Ein integrierter Ansatz zur interaktiven dreidimensionalen Simulation gekoppelter thermischer Prozesse

Von der
Fakultät Architektur, Bauingenieurwesen und Umweltwissenschaften
der Technischen Universität Carolo-Wilhelmina
zu Braunschweig

zur Erlangung des Grades eines
Doktoringenieurs (Dr.-Ing.)
genehmigte

Dissertation

von
Sebastian Bindick
geboren am 02. August 1981
aus Mettingen

Eingereicht am	28. Oktober 2010
Disputation am	16. Dezember 2010
Berichterstatter	Prof. Dr.-Ing. habil. Manfred Krafczyk
	Prof. Dr.-Ing. Markus König

2011

Vorwort und Danksagung

Die vorliegende Arbeit ist im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für rechnergestützte Modellierung im Bauingenieurwesen der Technischen Universität Braunschweig, sowie während meines Forschungsaufenthalts am Calit2 der University of California in San Diego, entstanden.

Mein ganz besonderer Dank gilt meinem Doktorvater Prof. Dr.-Ing. habil. M. Krafczyk, der mir mit seiner Anregung zu dieser Arbeit ein weites Betätigungsfeld eröffnete, das mir großen Freiraum für meinen Forschungsdrang ließ, mich gleichzeitig aber auch immer wieder zwang, mich auf das Wesentliche zu beschränken. Für den Einsatz als zweiter Berichterstatter danke ich Prof. Dr.-Ing. M. König. Ihm sowie Prof. Dr.-Ing. D. Dinkler und Prof. Dr.-Ing. D. Hosser danke ich für die freundliche Bereitschaft, die Aufgabe als Prüfer und den Vorsitz der Prüfungskommission zu übernehmen.

Danken möchte ich auch Prof. Dr. F. Küster für die Möglichkeit ein Forschungssemester bei ihm am Calit2 in San Diego zu verbringen, sowie dem DAAD für die finanzielle Förderung dieses Aufenthalts.

Die Unterstützung aller Mitarbeiter und Mitarbeiterinnen des Instituts für rechnergestützte Modellierung im Bauingenieurwesen war besonders wertvoll während der Erstellung der vorliegenden Arbeit. Ein besonderer Dank geht an meine Kollegen Christian Janßen und Maik Stiebler, die mir bei kleinen und größeren numerischen Problemstellungen zur Seite gestanden haben. Ein gleichzeitiger Dank geht auch an meinen Kollegen Jan Linxweiler für die vielen fruchtbaren Diskussionen zu Themen der Softwareentwicklung und seine Unterstützung bei der GPU-Programmierung.

Nicht zuletzt danke ich meinen Eltern, die mir mein Studium ermöglicht und mir beim Erreichen meiner Ziele immer Rückhalt gegeben haben.

Ganz speziell gilt mein Dank meiner Freundin Saskia. Sie hat unzählige Korrekturen durchgeführt und mir als wichtige Gesprächspartnerin gedient. Während der Zeit hat sie in besonderem Maße Verständnis und Geduld aufgebracht und mich in schwierigen Phasen immer wieder neu motiviert.

Braunschweig, Oktober 2010

Zusammenfassung

In der vorliegenden Arbeit werden neuartige Ansätze zur interaktiven Simulation thermischer Transportprozesse vorgestellt, wie sie für Ingenieuranwendungen insbesondere im Bauingenieurwesen typisch sind. Besonderer Fokus wird hierbei auf die Strahlungs-Struktur-Wechselwirkung gelegt, die für viele bauphysikalische Fragestellungen von großer Bedeutung ist.

Zur Lösung des Wärmestrahlungsproblems wird ein numerischer Ansatz basierend auf der hierarchischen Radiosity-Methode entwickelt, die den Strahlungsaustausch zwischen diffusen Oberflächen in einer abgeschlossenen Umgebung simuliert. Durch die Verwendung von optimierten Kd-Bäumen zur Speicherung der an der Strahlung beteiligten Objekte und der Verwendung eines adaptiven hierarchischen Ansatzes zur Berechnung des Strahlungsaustausches wird deren Komplexität deutlich reduziert. Die Strahlungssimulation ist direkt an die Berechnung der Wärmeleitung in der Struktur gekoppelt. Hierbei wird der Energietransport in wärmeleitenden Materialien für instationäre Temperaturfelder mit einem Finite-Differenzen-Verfahren (FDM) berechnet. Dieses wird aufgrund seiner besonderen Struktur effizient zur parallelen Berechnung auf Grafikkarten (GPGPUs) implementiert, um die typische Laufzeit um mehr als eine Größenordnung zu reduzieren.

Neben den modernen numerischen Ansätzen zur Lösung des physikalischen Problems werden auch Methoden des Computational Steering angewendet, die eine direkte Interaktion mit dem Simulationssystem zur Laufzeit (d.h. ohne die laufende Simulation zu unterbrechen) erlauben. Hierbei können innerhalb eines CAD-basierten virtuellen Entwurfsraumes komplexe Problemstellungen nicht nur transient simuliert werden, vielmehr es ist es möglich, das Systemverhalten interaktiv zu optimieren. Die Konstruktion des Gebäudemodells, sowie die Vorgabe von zusätzlichen bauteilspezifischen Parametern, wie Randbedingungen und Materialkennwerte, werden über das CAD-Werkzeug vorgegeben und können ebenfalls interaktiv verändert werden. Die Visualisierung der in jedem Zeitschritt anfallenden Simulationsergebnisse erfolgt verteilt innerhalb einer Tiled-Display-Umgebung bestehend aus vielen zusammengeschalteten Bilschirmen, die von einem Rendercluster angesteuert werden. Dieser verteilte Renderansatz erlaubt eine schnelle Ausgabe und Manipulation großer Datenmengen und stellt eine ideale Plattform für kooperative Planungsprozesse dar.

Der vorgestellte Prototyp wurde an Systemen, zu denen eine analytische Lösung existiert, validiert. Hierbei konnte gezeigt werden, dass die implementierten numerischen Verfahren für die Ankopplung der Strahlung an die Temperaturdynamik der Struktur asymptotisch die korrekte Lösung liefern. Außerdem zeigen mehrere bauphysikalische Anwendungsbeispiele mögliche Einsatzgebiete.

Abstract

In this thesis new approaches for interactive thermal simulations are presented which are applicable to several fields in civil engineering. For many problems in building physics heat transfer processes usually involve radiative heat transfer and heat conduction. For this reason the main focus of this work lies on the interaction of radiation and structure. To solve the complex radiative exchange between gray, diffuse surfaces in 3d domains an approach based on the hierarchical radiosity method is presented. The radiosity method is accelerated by using space partitioning techniques based on optimized kd-trees and an adaptive subdivision scheme of the surfaces. This approach decreases the complexity of the radiation problem considerably. The coupled transport of energy in heat conducting materials for transient temperature fields is calculated by a finite difference method. Since this approach requires substantial CPU time and memory, a GPU parallelization of the 3D finite difference scheme is implemented which accelerates the computational speed by more than one order of magnitude.

Furthermore computational steering techniques are applied, providing mechanisms for integrating modeling, simulation, data analysis, visualization and post-processing in a single environment including an interactive analyzation of the simulation results. Here a virtual interactive design space based on a CAD software is developed. Within such a system the user can interactively modify the geometry, boundary conditions and other parameters of the running simulation and explores the results immediately. For the large amounts of data processed during simulation paired with the requirement for immediate-mode and interactive visualization, a cluster-oriented rendering approach is presented. Here the simulation results are visualized on a tiled display system scaling to hundreds of mega pixels in resolution. This approach allows a group of planners and engineers to collaboratively optimize buildings at run-time with instantaneous updates to the simulation and visualization in a digital workspace.

In several validations it is shown that the presented software-prototype achieves high accuracy with only small deviations between analytical reference solutions and the simulation results. Finally some sample applications show the capability of this approach for complex scenarios in civil engineering.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

Symbolverzeichnis

Größe	Einheit	Bedeutung
a	m^2/s	Temperaturleitfähigkeit
α	–	Absorptionsgrad
α_{WF}	$W/(m^2K)$	Wärmeübergangskoeffizient
B	W/m^2	ausgehende Strahlungsstromdichte (Radiosity)
β	$^\circ$	Zenitwinkel
c_0	m/s	Lichtgeschwindigkeit
c	$kJ/(KgK)$	Spezifische Wärmekapazität
$d\omega$	–	Raumwinkel
Δt	–	Zeitschrittweite
$\Delta x, \Delta y, \Delta z$	–	Gitterabstand in die jeweilige Richtung
E	W/m^2	Eigenstrahlung
ε	–	Emissionsgrad
F_{ij}	–	Formfaktor, Sichtfaktor
h	$J \cdot s$	Planckkonstante
k	J/K	Boltzmannkonstante
L	W/m^2	Strahldichte
Λ	μm	Wellenlänge
λ	$W/(Km)$	Wärmeleitfähigkeit
$M(\Lambda, T)$	W	Spektrale spezifische Ausstrahlung
$M(T)$	W	Spezifische Ausstrahlung
ϕ	$^\circ$	Azimutwinkel
Φ	W/m^2	Strahlungsfluss
ϱ	kg/m^3	Dichte
ρ	–	Reflexionsgrad
σ	$W/(m^2\ K^4)$	Stefan-Boltzmann-Konstante
t	s	Zeit
T	$^\circ C, ^\circ K$	Temperatur
τ	–	Transmissionsgrad
$\dot{q}$	W/m^2	Wärmestromdichte, Wärmeflussdichte
$\dot{Q}$	W	Wärmestrom
$\dot{W}$	W/m^2	Leistungsdichte

Abkürzungsverzeichnis

AABB	Axis-Aligned Bounding Boxes
AEC	Architecture, Engineering und Construction
API	Application Programming Interface
ARX	AutoCAD Runtime Extension
ATC	Automatic Termination Criterion
BIM	Building Information Modeling
B-REP	Boundary Representation
BSP	Binary Space Partitioning
BVH	Bounding-Volume-Hierarchien
CAD	Computer Aided Design
CGLX	Cross-platform cluster Graphic Library
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DIN	Deutsche Industrie-Norm
FDM	Finite-Differenzen-Methode
FEM	Finite-Elemente-Methode
GPGPU	General Purpose Computation on Graphics Processing Unit
GPU	Graphics Processing Unit
IFC	Industry Foundation Classes
ISO	International Organization for Standardization
LBM	Lattice-Boltzmann-Methode
OBB	Oriented Bounding Box
OMF	Object Modeling Framework
RAM	Random-Access-Memory
SAH	Surface Area Heuristic
SIMD	Single Instruction Multiple Data
SM	Streaming Multiprozessor
SP	Streaming Prozessor
STEP	Standard for the Exchange of Product model data
UML	Unified Modeling Language
XML	Extensible Markup Language

1 Einleitung

1.1 Motivation

Für eine nachhaltige und energieeffiziente Planung von Gebäuden gewinnen detaillierte Kenntnisse über die zu erwartenden thermischen Verhältnisse in und um zu konstruierende Bauwerke zunehmend an Bedeutung. Neben einer optimalen Ausführung der wärmeschutztechnischen Maßnahmen an der Gebäudehülle, spielen Komfortbedürfnisse zur Schaffung eines behaglichen Raumklimas sowie ökologische Erwägungen zur Reduzierung der CO_2 Emissionen eine große Rolle. Eine optimale Ausführung von Gebäuden unter bauphysikalischen Aspekten kann nur unter Berücksichtigung der genauen Kenntnisse der zu erwartenden Temperaturverhältnisse erfolgen. Diese müssen bereits in der Entwurfsphase des Gebäudes berücksichtigt werden, so dass sich Baustoffe und Bauteile sinnvoll auswählen und dimensionieren, sowie Klima- und Heizungsanlagen optimieren lassen. Ein umfangreiches Wissen über komplexe thermische Prozesse in und an Gebäuden kann nur mit Hilfe von rechnergestützten transienten 3D Simulationen oder mittels extrem aufwändiger Experimente erlangt werden. Durch die Nutzung von Computersimulationen können Fachplaner und Ingenieure Fallstudien an komplexen Bauwerken durchführen, um so Erkenntnisse über das reale System zu erhalten, die bei der Gebäudeplanung und Optimierung helfen.

Vor diesem Hintergrund werden in dieser Arbeit Methoden zur effizienten interaktiven Simulation thermischer Transportprozesse entwickelt, wie sie für Ingenieuranwendungen insbesondere im Bauingenieurwesen typisch sind. Besonderer Fokus wird dabei auf die Strahlungs-Struktur-Wechselwirkung gelegt, die für viele bauphysikalische Fragestellungen von großer Bedeutung ist. Die Lösung des Wärmestrahlungsproblems basiert auf der Radiosity-Methode, einem numerischen Ansatz zur Berechnung des Strahlungsaustausches zwischen diffus strahlenden Oberflächen. Unter Verwendung von optimierten Kd-Bäumen zur Raumpartitionierung und adaptiven hierarchischen Methoden zur Strahlungsberechnung wird die Laufzeitkomplexität des Verfahrens deutlich reduziert. Die Simulation des gekoppelten diffusiven Wärmetransports in wärmeleitenden Materialien erfolgt mit einem effizienten hardwarebeschleunigten Finite-Differenzen-Ansatz. Hierdurch wird die typische Laufzeit um mehr als eine Größenordnungen reduziert. Als Datengrundlage für die thermischen Berechnungen dient eine hierarchische Datenstruktur für Gebäudemodelle auf Basis der Industry Foundation Classes (IFC), einem offenem Standard zur Beschreibung von Gebäudemodellen. Hierdurch wird ein einheitlicher Austausch von Planungsdaten zwischen verschiedenen Softwareprodukten im Bauwesen ermöglicht.

Neben den modernen algorithmischen Ansätzen zur Lösung des physikalischen Problems werden auch Methoden des Computational Steering eingesetzt, die eine direkte Modifikation des Simulationssystems zur Laufzeit (d.h. ohne die laufende Simulation zu unterbrechen) erlauben und es damit ermöglichen, ein schnelles und detaillierteres Wissen der Auswirkungen von Änderungen (beispielsweise an der Gebäudegeometrie) zu erlangen. Hierbei kann eine Gruppe von Fachplanern innerhalb eines virtuellen Entwurfsraumes kooperativ komplexe Gebäudestrukturen unter thermischen Gesichtspunkten optimieren.

1.2 Gliederung der Arbeit

In **Kapitel 2** werden die Grundlagen der Wärmeübertragung erläutert. Neben den physikalischen Größen und Gesetzen wird der Transport durch Wärmestrahlung und Wärmeleitung sowie ihre Kopplung im Detail betrachtet.

Kapitel 3 stellt die Entwicklung einer hierarchischen Datenstruktur für Gebäudemodelle auf Basis des Produktmodellstandards IFC vor. Durch diesen Ansatz ist der einheitliche Austausch komplexer Planungsdaten (für thermische Simulationen) zwischen verschiedenen Softwareprodukten im Bauwesen möglich. Grundlage der Datenstruktur bilden optimierte Kd-Bäume zur Raumpartitionierung durch die sich die Algorithmen zur thermischen Simulation besonders effizient ausführen lassen.

Kapitel 4 beschreibt einen effizienten numerischen Lösungsansatz zur verteilten Simulation des gekoppelten zeitabhängigen Problems von Wärmeleitung und Wärmestrahlung auf Basis eines adaptiven Radiosity-Ansatzes und einer hardware-beschleunigten Finite-Differenzen-Methode.

In **Kapitel 5** wird die Entwicklung einer Simulationsumgebung vorgestellt, die eine interaktive Steuerung und Visualisierung der laufenden Simulation ermöglicht. Als Entwurfsraum zur Konstruktion komplexer Gebäudemodellen dient hierbei die CAD-Software AutoCAD Architecture, welche direkt an die Simulation gekoppelt ist. Außerdem werden Verfahren zur verteilten Visualisierung auf Computerclustern und die Darstellung innerhalb von Tiled-Display-Umgebungen erläutert.

Abschließend wird in **Kapitel 6** der Simulationskern anhand von analytischen Referenzlösungen validiert sowie einige komplexe Anwendungsbeispiele aus dem Bau- und Umweltingenieurwesen betrachtet.

2 Grundlagen der Wärmeübertragung

Wärmeübertragung (heat transfer) ist ein Teilgebiet der Thermodynamik, das sich mit der Beschreibung von Wärmetransportvorgängen befasst. Diese spielen in vielen Bereichen des Ingenieurwesens und der Naturwissenschaften (z.B. dem Bauingenieurwesen, den Umweltwissenschaften, der Energietechnik oder dem Maschinenbau) eine große Rolle. Die auftretenden Phänomene erstrecken sich über unterschiedliche räumliche und zeitliche Skalen beginnend bei der Materialoptimierung (z.B. von Asphalt oder Dämmstoffen) unter thermischen Gesichtspunkten, über wärmeschutztechnische Maßnahmen an der Gebäudehülle, bis hin zu Problemen der globalen Erwärmung. Durch die Untersuchung thermischer Transportvorgänge lassen sich Bauteile sinnvoll dimensionieren, der Einsatz von Energie minimieren, eine Maximierung oder Minimierung (je nach Zielsetzung) der übertragenen Wärmemenge erreichen oder Konstruktionen thermisch optimieren. Häufig auftretende Fragestellungen befassen sich hierbei mit den zu erwartenden Temperaturen und Leistungsdichten, den beeinflussenden Mechanismen und Größen des Wärmetransports sowie dem zeitlichen Ablauf der Wärmeübertragung [95]. Dieses Kapitel behandelt zunächst die grundlegenden physikalischen Größen und Gesetze des Wärmetransports. Anschließend wird die Wärmeübertragung durch Strahlung und Wärmeleitung im Detail erläutert.

2.1 Begriffe, Größen, Transportgesetze

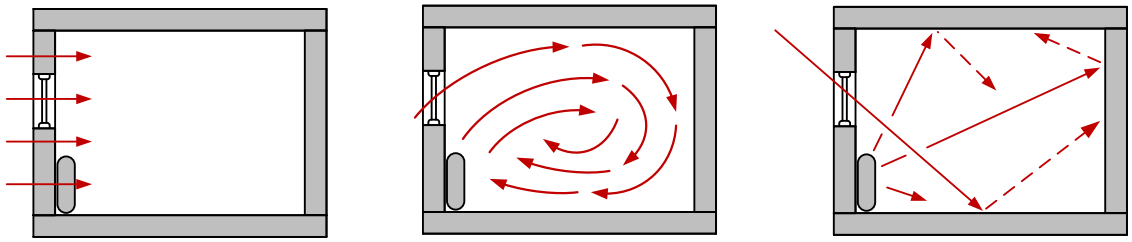
Bei der Wärmeübertragung wird bedingt durch einen Temperaturunterschied thermische Energie, die als Wärme (heat) bezeichnet wird, übertragen [9, 95, 122, 39]. Die pro Zeiteinheit übertragene Wärmemenge bezeichnet man als Wärmestrom (heat flux) $\dot{Q}$, welcher mit einer Entropieänderung verbunden ist:

$$dQ = T\,dS \quad \left[\frac{J}{s} = W\right]. \tag{2.1}$$

Hierbei fließt nach dem 2. Hauptsatz der Thermodynamik Wärme immer in Richtung fallender Temperatur. Damit verbunden ist ein thermischer Ausgleich über die Systemgrenzen hinweg. Die Intensität des Wärmestroms wird als Wärmestromdichte oder Wärmeflussdichte (heat flux density) $\dot{q}$ bezeichnet, welche den Wärmetransport pro Zeit- und Flächeneinheit beschreibt:

$$\dot{q} = \frac{Q}{A\Delta t} = \frac{\dot{Q}}{A} \quad \left[\frac{J}{m^2 s} = \frac{W}{m^2}\right]. \tag{2.2}$$

An der Wärmeübertragung sind drei unterschiedliche Transportphänomene beteiligt: **Wärmeleitung**, **konvektive Wärmeübertragung** und **Wärmestrahlung**, diese können allein oder in Kombination miteinander auftreten (vgl. Abbildung 2.1).



(a) *Wärmeleitung* ***(b)*** *Konvektion* ***(c)*** *Strahlung*

Abbildung 2.1: *Arten der Wärmeübertragung*

Bei der **Wärmeleitung** fließt thermische Energie (Wärme) in einem Feststoff oder ruhenden Fluid infolge eines Temperaturgradienten und ist nicht mit dem Transport von Teilchen verbunden. Ein wichtiges Maß für den Wärmestrom in einem Körper ist die Wärmeleitfähigkeit des Materials. **Konvektion** beschreibt den Transport von Wärme in einem Fluid aufgrund von Teilchenbewegungen und tritt in Gasen und Flüssigkeiten auf. Erst durch die Strömung eines Fluids wird Konvektion möglich. **Wärmestrahlung** ist eine elektromagnetische Strahlung die von jedem Körper mit einer Temperatur oberhalb des absoluten Nullpunktes emittiert wird. Wärmeübertragung durch Strahlung ist nicht an Materie gebunden.

Im Rahmen dieser Arbeit wird besonderer Fokus auf die Simulation gekoppelter Strahlungs- und Wärmeleitungsprozesse gelegt, der konvektive Transport wird hier nicht weiter behandelt.

2.2 Wärmestrahlung

Jeder Körper mit einer Temperatur oberhalb des absoluten Nullpunktes emittiert ein kontinuierliches Spektrum an elektromagnetischen Wellen [9, 34, 61, 79, 105, 122]. Diese Art der Wärmeabgabe wird als Wärmestrahlung, thermische Strahlung oder Temperaturstrahlung bezeichnet und erfordert im Gegensatz zur Wärmeleitung oder Konvektion keinen stofflichen Träger. Bei der Wärmeübertragung durch Strahlung wird durch Emissions- und Absorptionsprozesse Wärmeenergie zwischen voneinander getrennten Körpern ausgetauscht. Bei der Emission wird die innere Energie eines Körpers in Energie umgewandelt, die durch elektromagnetische Wellen transportiert wird. Hierbei stammt die ausgehende Strahlung von einer dünnen Oberflächenschicht (ca. $1\mu m$). Aus diesem Grund spricht man in der Regel von strahlenden Flächen und nicht von strahlenden Körpern. Die Intensität der Strahlung hängt von der Oberflächenbeschaffenheit und der Temperatur des Körpers ab. Treffen elektromagnetische Wellen auf einen Körper, wird ein Teil der Strahlungsenergie absorbiert

und in Wärme umgewandelt, während der Rest reflektiert oder transmittiert wird (vgl. Abbildung 2.2). Diese Anteile sind abhängig von den Materialeigenschaften und werden als Absorptionsgrad α, Reflexionsgrad ρ und Transmissionsgrad τ bezeichnet und stehen in folgender Relation zu einander:

$$\alpha + \tau + \rho = 1. \tag{2.3}$$

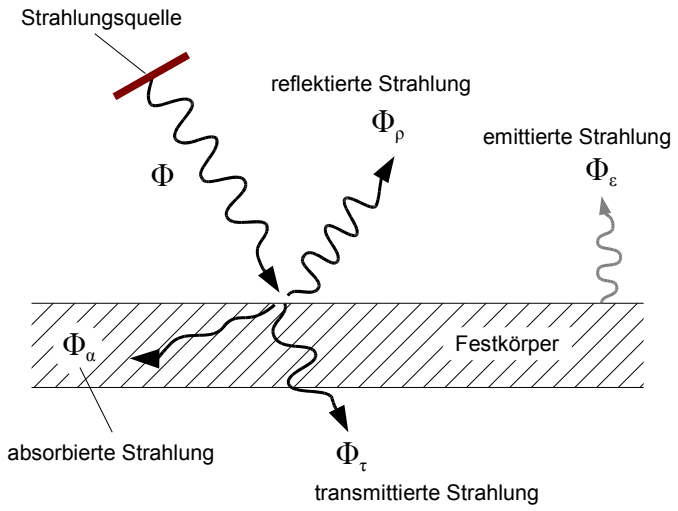


Abbildung 2.2: *Reflexion, Absorption und Transmission [95]*

Elektromagnetische Wellen breiten sich im Vakuum mit Lichtgeschwindigkeit aus, wodurch Wärme durch Strahlung über große Entfernung ausgetauscht werden kann. Ein Beispiel dafür ist der große Energiestrom zwischen Sonne und Erde. Die Energiedichte der Wärmestrahlung ist nicht gleichmäßig über das gesamte Spektrum der elektromagnetischen Wellen verteilt (vgl. Abbildung 2.3). Aus wärmetechnischer Sicht ist der Wellenlängenbereich zwischen $0,1\mu m$ und $100\mu m$ interessant. In diesem Bereich strahlen Körper mit Temperaturen zwischen wenigen Kelvin und $2 \cdot 10^4$ Kelvin.

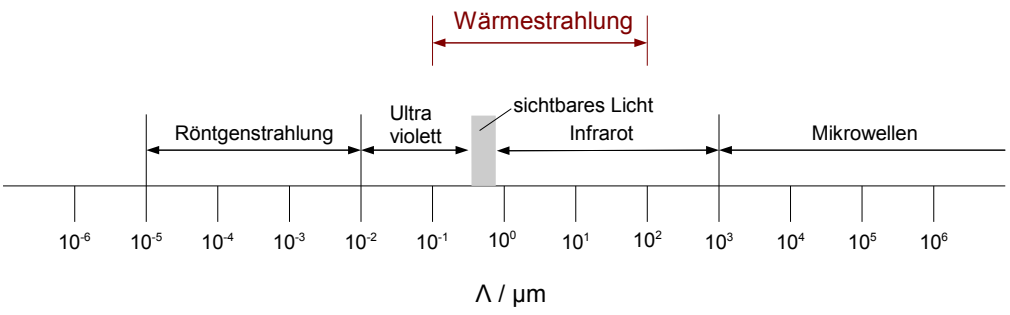


Abbildung 2.3: *Spektrum der elektromagnetischen Wellen [9]*

Zur theoretischen Beschreibung von Wärmestrahlung können zwei unterschiedliche Ansätze verwendet werden: die klassische Theorie der elektromagnetischen Wellen und die Quantentheorie der Photonen (Lichtteilchen) [105]. Zur Beschreibung des Strahlungsaustausches zwischen Oberflächen ohne Teilnahme des dazwischenliegenden Mediums (Gasstrahlung) hat sich die Theorie der elektromagnetischen Wellen etabliert. Aus diesem Grund wird auf die quantentheorethischen Ansätze in dieser Arbeit nicht weiter eingegangen.

Die folgenden Abschnitte geben einen Überblick über die Eigenschaften und Gesetze von Wärmestrahlung. Eine detailliertere Beschreibung der strahlungsphysikalischen Grundlagen kann Baehr [9], der DIN 9288 [34], Modest [79] und Siegel [105] entnommen werden.

2.2.1 Strahlungsphysikalische Größen

Strahlung ist nicht nur von der Wellenlänge abhängig sondern auch von der Verteilung auf die Richtungen im Raum. Diese doppelte Abhängigkeit macht die Betrachtung von Wärmestrahlung kompliziert, weshalb vier unterschiedliche Arten von physikalischen Größen benötigt werden [9]:

- *Gerichtete spektrale Größen* beschreiben Strahlung in Abhängigkeit der Wellenlänge und der Verteilung auf die einzelnen Richtungen des Raums. Da sie schwierig zu erfassen und berechnen sind, werden häufig Größen verwendet, die nur einen der beiden Effekte erfassen.
- *Hemisphärische spektrale Größen* hängen nur von der Wellenlänge ab und fassen die Strahlung über alle Richtungen des Raums zusammen.
- *Gerichtete Gesamtgrößen* beschreiben die Strahlung über die Richtungen des Raums und fassen die Wellenlängen zusammen.
- *Hemisphärische Gesamtgrößen* fassen sowohl die Wellenlängen als auch die Raumrichtungen der Strahlung zusammen und stellen somit die einfachste Form der strahlungsphysikalischen Größen dar. Zur Lösung vieler Ingenieurprobleme sind sie häufig ausreichend.

Die wichtigsten Größen werden im Folgenden kurz erläutert.

Spektrale Strahldichte

Zur Beschreibung des von einer Fläche ausgehenden Strahlungsflusses $d^3\Phi$ wird eine Verteilungsfunktion, die sogenannte spektrale Strahldichte $L(\Lambda, \beta, \phi, T)$ eingeführt [9, 79, 105]. Sie ist eine gerichtete spektrale Größe in Abhängigkeit der Oberflächenbeschaffenheiten eines Körpers und gibt an, welcher Strahlungsfluss von einem Punkt der Oberfläche in ein Wellenlängenbereich Λ und die Raumrichtung (β, ϕ) bei einer Temperatur T emittiert wird. Um die Strahlungsintensität auf die einzelnen Richtungen des Raums abzubilden werden zwei

Winkelkoordinaten, der Zenitwinkel β (ausgehend von der Flächennormalen) und der Azimutwinkel φ, festgelegt (vgl. Abbildung 2.4). Der Strahlungsfluss $d\Phi$ auf eine Fläche dA_n im Abstand r ist proportional zum Raumwinkelelement $d\omega$ (vgl. Gleichung 2.4).

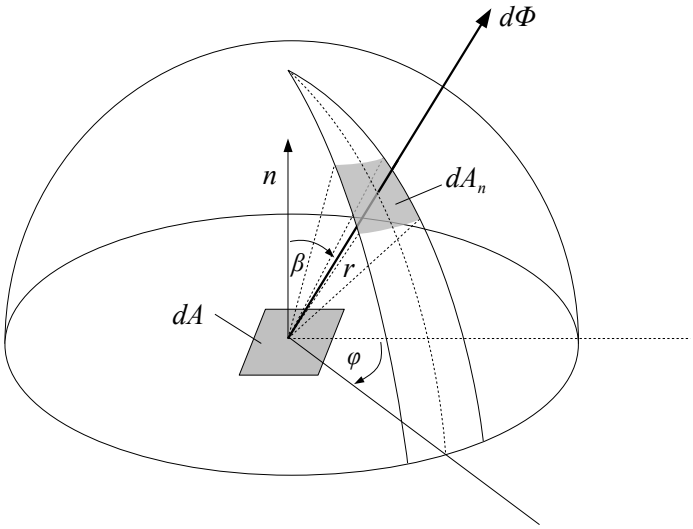


Abbildung 2.4: *Ausgehender Strahlungsfluss $d\Phi$ in ein infinitesimales Raumwinkelelement in Richtung des Zenitwinkels β und des Azimutwinkels φ [122]*

Anmerkung: *Der **Raumwinkel** (solid angle) ist ein dreidimensionaler Winkel zwischen zwei Vektoren der proportional zur Oberfläche der Einheitskugel ist. Für eine beliebige Fläche dA ergibt sich der Raumwinkel $d\omega$ durch Projektion von dA auf eine Kugel vom Radius r:*

$$d\omega = \frac{dA_n}{r^2}. \tag{2.4}$$

Der volle Raumwinkel beträgt 4π. Die Einheit des Raumwinkels ist dimensionslos, dennoch wird in der Regel die Einheit Steradiant verwendet, die dem Bogenmaß beim ebenen Winkel in der Einheit Radiant entspricht.

Somit berechnet sich der ausgehende Strahlungsfluss $d^3\Phi$ von einem Punkt auf dem Flächenelement dA in einem infinitesimalen Raumwinkelelement $d\omega$ und innerhalb eines Wellenlängenintervalls $d\Lambda$ mit:

$$d^3\Phi = L(\Lambda, \beta, \phi, T)\ \cos\beta\ dA\ d\omega\ d\Lambda \quad \left[\frac{W}{m^2\ \mu m\ sr}\right]. \tag{2.5}$$

Die Strahldichte wird hierbei auf das in Abstrahlrichtung projizierte Flächenelement $\cos\beta\ dA$ bezogen und nicht wie üblich auf die abstrahlende Fläche dA (vgl. Abbildung 2.5). Die abgegebene Strahlungsleistung hängt von den richtungsabhängigen physikalischen Strahlungseigenschaften der Oberfläche, sowie von der Projektion des Flächenelements

senkrecht zur Strahlrichtung ab. Diese Projektion ist aus rein geometrischen Überlegungen notwendig, da die Ausstrahlung für $\beta = \pi/2$ null wird und ihr Maximum bei $\beta = 0$ erreicht. Die unter dem Zenitwinkel β abgegebene Strahlungsfluss ist um den Faktor $\cos\beta$ geringer als die senkrecht abgegebene Leistung.

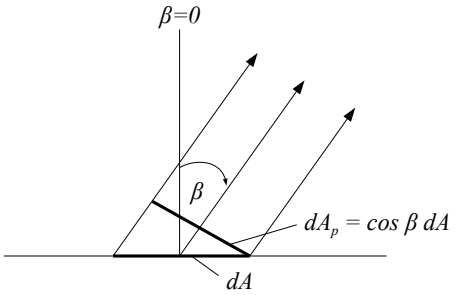


Abbildung 2.5: *Projektion des Flächenelements senkrecht zur Strahlrichtung [9]*

Da die spektrale Strahldichte schwierig zu erfassen und zu berechnen ist, wird häufig eine Strahldichte verwendet, die nur einen der beiden Effekte (die Richtungsabhängigkeit oder die Frequenzabhängigkeit) erfasst.

Spektrale spezifische Ausstrahlung

Werden die richtungsabhängigen physikalischen Strahlungseigenschaften der Oberfläche vernachlässigt, was in vielen Fällen eine brauchbare Näherung ist (siehe hierzu Abschnitt 2.2.4), spricht man von diffus strahlenden Körpern [9, 79, 105]. Diese auch als Lambert-Strahler bezeichneten Körper geben in alle Raumrichtungen eine konstante Strahldichte ab. Hierbei wird ausschließlich der im vorhergehenden Abschnitt erläuterte geometrische Effekt (durch Projektion des Flächenelements dA) berücksichtigt (vgl. Abbildung 2.5). Eine weitere Winkelabhängigkeit gibt es nicht.

Anmerkung: *Ein **Lambert-Strahler** ist ein physikalisch halbideal strahlender Körper, der in alle Richtungen des Halbraums die gleiche Strahldichte abgibt. Der eintreffende Strahl wird hierbei vollständig diffus reflektiert, das heißt Reflexion und Emission sind unabhängig von der Abstrahlrichtung. Der nach Johann Heinrich Lambert benannte Körper stellt für reale Körper oft eine gute Näherung dar [9, 79].*

Für eine ortsunabhängige Strahldichte $L(\Lambda, T)$ vereinfacht sich der Strahlungsfluss (Gleichung 2.5) zu:

$$d^2\Phi = L(\Lambda, T)\ \cos\beta\ d\omega\ d\Lambda. \tag{2.6}$$

Soll die spektrale spezifische Ausstrahlung $M(\Lambda, T)$ von einem endlich großen Flächenelement berechnet werden, so muss über den Raumwinkel $d\omega$ integriert werden. Hieraus folgt die hemisphärische Größe:

$$\boxed{M(\Lambda, T) = \int_{\omega} L(\Lambda, T)\ \cos\beta\ d\omega\ d\Lambda.} \tag{2.7}$$

Die spektrale spezifische Ausstrahlung eines schwarzen Körpers ist durch das Plancksche Strahlungsgesetz (vgl. Gleichung 2.9) gegeben [9, 105].

Spezifische Ausstrahlung

Betrachtet man Wärmestrahlung über einen konstanten Frequenzbereich, so vereinfacht sich (Gleichung 2.7) durch Integration über alle Wellenlängen zu [9, 105]:

$$\boxed{M(T) = L(T) \int_{\omega} \cos\beta\ d\omega} \tag{2.8}$$

Diese hemisphärische Gesamtgröße wird als spezifische Ausstrahlung bezeichnet. Man beachte die nun konstante Strahldichte $L(T)$, die vor das Integral gezogen werden kann. Das Integral kann unabhängig von der Strahldichte gelöst werden und hängt nur noch von dem Raumwinkel ab, der sich aus der Lage und Orientierung zweier aufeinander abstrahlender Flächen ergibt. Die spezifische Ausstrahlung ist durch das Stefan-Boltzmann-Gesetz (vgl. Gleichung 2.10) gegeben.

2.2.2 Emission von Strahlung

Ein Körper mit einer Temperatur größer als null Kelvin emittiert einen kontinuierlichen Energiestrom durch Umwandeln seiner inneren thermischen Energie in elektromagnetische Wellen [9, 105]. Diese vom Körper ausgehende spektrale spezifische Ausstrahlung $M(\Lambda, T)$ eines idealen schwarzen Körpers lässt sich mit dem Planckschen Strahlungsgesetz berechnen:

$$M(\Lambda, T) = \frac{2\pi c_0^2}{h^4} \frac{A}{e^{(hc_0/k\Lambda T)} - 1} d\Lambda \quad [W]. \tag{2.9}$$

Das von Max Planck im Jahre 1900 hergeleitete Strahlungsgesetz beschreibt die Verteilung der Energie auf die einzelnen Wellenlängen Λ abhängig von der absoluten Temperatur T, der Planckkonstante $h = 6{,}626 \cdot 10^{-34}\ J \cdot s$, der Lichtgeschwindigkeit c_0 und der Boltzmannkonstante $k = 1{,}381 \cdot 10^{-23}\ J/K$. Diese Verteilung ist nicht gleichmäßig über das gesamte Spektrum. Sie steigt mit zunehmender Wellenlänge bis zu einem Maximalwert bei Λ_{max} an (vgl. Abbildung 2.6).

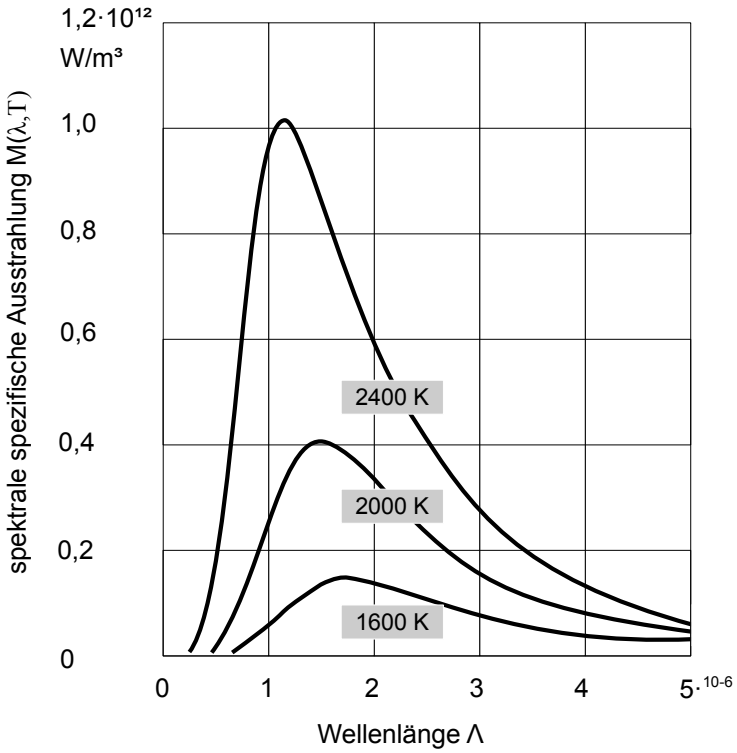


Abbildung 2.6: *Spektrale spezifische Ausstrahlung*

Anmerkung: *Ein **schwarzer Körper** ist ein idealisierter Körper, der als Grundlage für viele theoretische und praktische Betrachtungen dient. Der Begriff wurde 1860 von dem Physiker Gustav Kirchhoff geprägt. Die auf den schwarzen Körper einfallende Strahlung wird vollständig absorbiert ($\alpha = 1.0$), d.h. es wird weder Strahlung reflektiert ($\rho = 0.0$) noch transmittiert ($\tau = 0.0$). Außerdem stellt ein schwarzer Körper eine ideale Strahlungsquelle ($\varepsilon = 1.0$) dar, bei der die höchstmögliche Energiemenge bei einer Temperatur T abgestrahlt wird.*

Durch Integration über alle Wellenlängen folgt aus Gleichung 2.9 die hemisphärische Gesamtgröße, die als Stefan-Boltzmann-Gesetz bezeichnet wird [79]:

$$M(T) = \varepsilon\, \sigma\, A\, T^4 \quad \left[\frac{W}{m^2}\right], \tag{2.10}$$

mit der Stefan-Boltzmann-Konstante:

$$\sigma = 5,670 \cdot 10^{-8} \quad \frac{W}{m^2\, K^4} \tag{2.11}$$

und dem Emissionsgrad ε, der beschreibt wie viel Strahlung ein Körper im Vergleich zu einem idealen schwarzen Strahler abgibt. Das Stefan-Boltzmann-Gesetz ist nach den Physikern Josef Stefan und Ludwig Boltzmann benannt, die es im Jahr 1879 entwickelt haben.

Erst einige Jahre später, um 1900, konnte Max Planck mit seinem Strahlungsgesetz (vgl. Gleichung 2.9) unter Berücksichtigung der Quantentheorie, das Stefan-Boltzmann-Gesetz herleiten und in Zusammenhang mit anderen thermodynamischen Überlegungen bringen.

Das Plancksche Strahlungsgesetz und das Stefan-Boltzmann-Gesetz sind die beiden wichtigsten Gesetze zur Betrachtung von Strahlung.

2.2.3 Strahlungsaustausch

Bei der Wärmeübertragung durch Strahlung wird Energie nicht nur von dem wärmeren an den kälteren Körper Energie abgegeben, sondern es wird auch Energie vom kälteren an den wärmeren Körper übertragen [9, 79, 105]. Dieser sogenannte Strahlungsaustausch hängt im Wesentlichen von den Oberflächenbeschaffenheiten sowie der Position und Orientierung der Flächen ab. Außerdem hat das Zwischenmedium, welches die Flächen voneinander trennt, Einfluss auf den Strahlungsaustausch. Das Medium wirkt hierbei absorbierend, emittierend und streuend auf die Strahlung. Dieser Prozess wird als Gasstrahlung bezeichnet und tritt häufig bei Verbrennungsprozessen auf, bei denen sich Gase mit Rußpartikeln vermischen oder besondere Gasgemische verwendet werden. Die Betrachtung von Gasstrahlung ist schwierig, da Gase nicht in erster Näherung grauen Körpern entsprechen. Eine ausführliche Beschreibung zur Modellierung von Gasstrahlung kann u.a. den Arbeiten von Hottel et al. [56] und Siegel et al. [105] entnommen werden. Im Rahmen dieser Arbeit wird der Einfluss des Zwischenmediums vernachlässigt, da viele Gase (wie z.B. Luft), die in Verbindung mit niedrigen bis mittleren Temperaturen in typischen Anwendungen des Bauingenieurwesens auftreten (mit Ausnahme der Rauchgasentwicklung bei Bränden), den Strahlungsaustausch nur in geringem Maße beeinflussen.

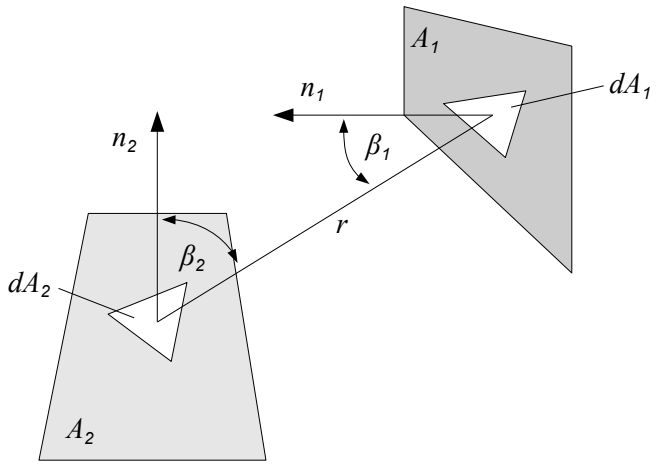


Abbildung 2.7: *Größen zur Formfaktorberechnung*

In diesem Abschnitt wird der Transport von Strahlungsenergie zwischen Oberflächen mit idealisierten diffusen Materialeigenschaften betrachtet. Strahlungsaustausch bedeutet, dass

jede Fläche einer Umgebung mit jeder anderen Fläche in Interaktion steht, d.h. das Strahlungsenergie zwischen ihnen ausgetauscht wird, unabhängig davon wie weit die beiden Flächen voneinander entfernt sind. Die Menge der ausgetauschten Energie hängt hierbei im Wesentlichen von Größe, Distanz und Orientierung der Flächen ab und führt zu einer geometrischen Funktion, dem sogenannten Formfaktor. Der Formfaktor, teilweise auch als Sichtfaktor (form factor, view factor oder configuration factor) bezeichnet, beschreibt den Anteil der abgegebenen Energie einer Fläche auf eine andere Fläche und liegt zwischen 0.0 (keine Energie wird abgegeben) und 1.0 (die Energie wird vollständig abgegeben). Die im Formfaktor auftauchenden Größen sind in Abbildung 2.7 dargestellt.

Zur Berechnung des Formfaktors wird der Strahlungsfluss nach Gleichung 2.6 unter Annahme ideal diffuser Oberflächen betrachtet [9]. Für den Strahlungsfluss ausgehend von der Fläche dA_1 mit einer Strahldichte L_1 zur Fläche dA_2 erhält man folgende Gleichung:

$$d^2\Phi_{12} = L_1 \cos\beta_1 \, dA_1 \, d\omega_2. \tag{2.12}$$

Der von dA_1 betrachtete Raumwinkel zur Fläche dA_2 ergibt sich aus Gleichung 2.4 zu:

$$d\omega_2 = \frac{\cos\beta_2 \, dA_2}{r^2}. \tag{2.13}$$

Durch Einsetzen des Raumwinkels in Gleichung 2.12 folgt der Strahlungsfluss, der ausgehend von Fläche dA_1 die Fläche dA_2 erreicht:

$$d^2\Phi_{12} = L_1 \frac{\cos\beta_1 \cos\beta_2}{r^2} \, dA_1 \, dA_2. \tag{2.14}$$

Diese Gleichung ist als fotometrisches Grundgesetz bekannt. Der übertragene Wärmefluss nimmt hierbei mit dem Quadrat des Abstands r und dem Kosinus kleiner werdender Zenitwinkel (β_1 und β_2) ab.

Integriert man das fotometrische Grundgesetz über beide Flächen, erhält man den Strahlungsfluss für endliche Flächen dA_1 und dA_2 unter Annahme konstanter Strahldichte L_1:

$$\Phi_{12} = L_1 \int_{A_1}\int_{A_2} \frac{\cos\beta_1 \cos\beta_2}{r^2} \, dA_1 \, dA_2 \tag{2.15}$$

Hieraus folgt mit dem Strahlungsfluss $\Phi_1 = \pi L_1 A_1$, der von der Fläche dA_1 in den Raum abgegeben wird, für den Formfaktor:

$$\boxed{F_{12} = \frac{\Phi_{12}}{\Phi_1} = \frac{1}{\pi A_1} \int_{A_1}\int_{A_2} \frac{\cos\beta_1 \cos\beta_2}{r^2} \, dA_1 \, dA_2.} \tag{2.16}$$

Der so ermittelte allgemeine Formfaktor hängt nur noch von der Sender- und Empfängergeometrie ab, berücksichtigt also die Ausrichtung und den Abstand der Teilflächen zueinander. Der Formfaktor gilt nur für diffus strahlende Flächen unter Annahme homogener Oberflächeneigenschaften (konstante Temperatur und Strahldichte) über die einzelnen Flächenelemente. Die von einer Fläche dA_1 an eine Fläche dA_2 übertragene Strahlungsenergie

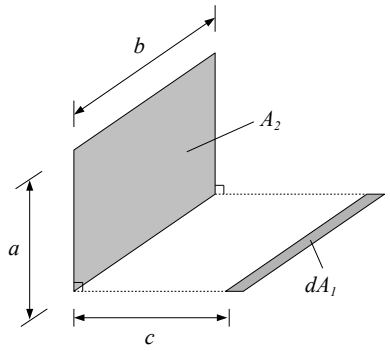
verhält sich reziprok zu der Energie die von Fläche dA_2 an Fläche dA_1 ausgetauscht wird. Hieraus folgt die Beziehung:

$$A_1 F_{12} = A_2 F_{21}. \tag{2.17}$$

Diese Eigenschaft spielt eine große Rolle, da für den Strahlungsaustausch zwischen zwei Flächen das Doppelintegral nur einmal gelöst werden muss.

Das Formfaktorintegral (Gleichung 2.16) wurde für zahlreiche einfache geometrische Anordnungen berechnet. Einige dieser Lösungen sind in Tabelle 2.1 und Tabelle 2.2 angegeben [57]. Ausführlichere Sammlungen von berechneten Formfaktoren können u.a. Howell [57], Modest [79] oder Siegel et al. [105] entnommen werden.

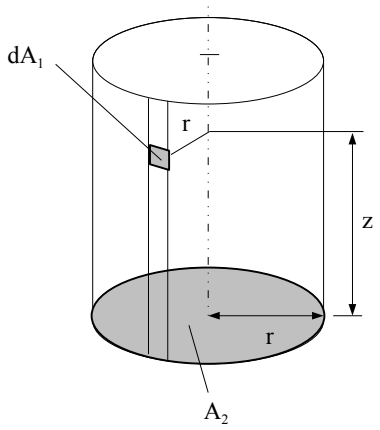
Schmaler Streifen und Rechteck im 90° Winkel



$$X = a/b, \quad Y = c/b$$

$$F_{ij} = \frac{1}{\pi}\left[\tan^{-1}\frac{1}{Y} + \frac{Y}{2}\ln\frac{Y^2(X^2+Y^2+1)}{(Y^2+1)(X^2+Y^2)} - \frac{Y}{\sqrt{X^2+Y^2}}\tan^{-1}\frac{1}{\sqrt{X^2+Y^2}}\right]$$

Innen liegendes infinitesimales Element eines Zylinders und ein Zylinderboden

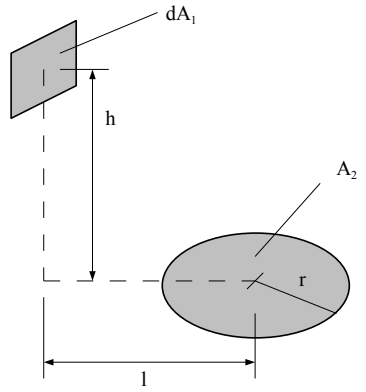


$$Z = z/r$$

$$F_{ij} = \frac{Z^2+2}{2\sqrt{Z^2+4}} - \frac{Z}{2}$$

Tabelle 2.1: *Formfaktoren für unterschiedliche Geometrien (Teil 1) aus [57]*

Infinitesimales Element und eine Kreisscheibe

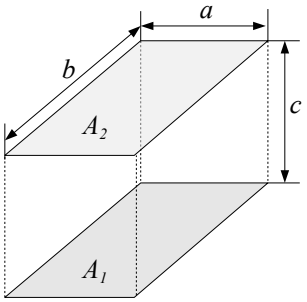


$$H = h/l, \quad R = r/l$$

$$Z = 1 + H^2 + R^2$$

$$F_{ij} = \frac{H}{2}\left[\frac{Z}{\sqrt{Z^2 - 4R^2}} - 1\right]$$

Zwei parallele, gegenüberliegende Rechtecke



$$X = a/c, \quad Y = b/c$$

$$F_{ij} = \frac{2}{\pi XY}\left\{\ln\left[\frac{(1+X^2)(1+Y^2)}{1+X^2+Y^2}\right]^{1/2}\right.$$

$$+X\sqrt{1+Y^2}\tan^{-1}\frac{X}{\sqrt{1+Y^2}} + Y\sqrt{1+X^2}$$

$$\left.\cdot\tan^{-1}\frac{Y}{\sqrt{1+X^2}} - X\tan^{-1}X - Y\tan^{-1}Y\right\}$$

Tabelle 2.2: *Formfaktoren für unterschiedliche Geometrien (Teil 2) aus [57]*

2.2.4 Strahlungseigenschaften realer Körper

Im Folgenden werden die strahlungsphysikalischen Materialeigenschaften realer Körper aufgezeigt. Die in den vorherigen Abschnitten betrachteten Eigenschaften von idealen schwarzen Körpern unterscheiden sich stark von denen realer Oberflächen [9, 79, 105]. So emittiert ein schwarzer Körper die physikalisch größtmögliche Strahlungsleistung in Abhängigkeit seiner Oberflächentemperatur. Diese Annahme trifft auf reale Körper nicht zu, ihr abgegebener Strahlungsfluss erreicht nie die höchstmögliche Energiemenge. Die tatsächliche abgegebene Strahlungsintensität wird durch den sogenannten Emissionsgrad ε beschrieben, der im Bereich $0.0 < \varepsilon < 1.0$ liegt. Der Emissionsgrad ist das Verhältnis der von einem realen Körper zu der von einem schwarzen Körper emittierten Strahlung. Ein Körper mit dem Emissionsgrad von $\varepsilon = 0.0$ wird als ideal weißer Körper und ein Körper mit

$\varepsilon = 1.0$ als idealer schwarzer Körper bezeichnet. Analog hierzu wird das Vermögen eines Körpers, Strahlungsenergie aufzunehmen, durch den Absorptionsgrad α ausgedrückt. Der Zusammenhang zwischen Emission und Absorption wurde von dem deutschen Physiker Gustav Robert Kirchhoff entdeckt. Das Kirchhoffsche Strahlungsgesetz besagt, dass der Emissionsgrad und der Absorptionsgrad identisch zueinander sind:

$$\varepsilon(\Lambda, \beta, \varphi, T) = \alpha(\Lambda, \beta, \varphi, T). \tag{2.18}$$

Das bedeutet, dass ein Körper im gleichen Maße Wärme abgeben wie aufnehmen kann. Auch das Strahlungsreflexionsvermögen (Reflexionsgrad ρ) einer Oberfläche lässt sich auf den Emissionsgrad zurückführen:

$$\rho(\Lambda, \beta, \varphi, T) = 1.0 - \varepsilon(\Lambda, \beta, \varphi, T). \tag{2.19}$$

Somit ist nach Kirchhoff nur die Materialfunktion $\varepsilon(\Lambda, \beta, \varphi, T)$ zur Beschreibung der Strahlungseigenschaften opaker (undurchlässiger) Körper notwendig.

Emissionsgrade realer Körper ändern sich in Abhängigkeit der Temperatur, der Frequenzen sowie der verschiedenen Ausstrahlrichtungen. Das bedeutet, dass ein Körper je nach Frequenzbereich und Ausstrahlrichtung unterschiedlich viel Energie abgibt. Für eine vollständige Beschreibung der Oberflächen ist der Emissionsgrad als Funktion $\varepsilon(\Lambda, \beta, \varphi, T)$ der Frequenz, der Ausstrahlwinkel und der Temperatur anzugeben. Die ausgehende Strahldichte L einer realen Oberfläche ändert sich somit zu:

$$L_\Lambda(\Lambda, \beta, \varphi, T) = \varepsilon_\Lambda(\Lambda, \beta, \varphi, T) L(\Lambda, T). \tag{2.20}$$

Durch die komplizierten Abhängigkeiten der Materialfunktion ist die experimentelle Bestimmung des Emissionsgrads für verschiedene Stoffe äußerst schwierig. Aus diesem Grund werden häufig Näherungen getroffen, welche die Ermittelung der Materialeigenschaften sowie die Strahlungsberechnung vereinfachen. Eine brauchbare Approximation für reale Körper ist die Annahme von diffus strahlenden Oberflächen (Lambert Strahler) mit einem richtungsunabhängigem Emissionsgrad $\varepsilon(\Lambda, T)$.

Eine weitere Vereinfachung ist die Annahme eines grauen Strahlers, bei der die ausgehende Strahlungsstromdichte eines Körpers proportional zu der eines schwarzen Körpers mit gleicher Temperatur ist (vgl. Abbildung 2.8). In diesem Fall ist der Emissionsgrad $\varepsilon(T)$ von der Wellenlänge unabhängig. Diese Annahme triff jedoch nicht in allen Fällen zu, da der Emissionsgrad bei realen Körpern merklich von der Wellenlänge abhängt; insbesondere wenn Solarstrahlung berücksichtigt werden soll. Für viele Materialien kann die Temperaturabhängigkeit auch in großen Temperaturbereichen vernachlässigt werden, da sich ε kaum ändert. Eine Ausnahme stellen hier die metallischen Oberflächen dar, bei denen bei tiefen Temperaturen der Emissionsgrad fast proportional zur Temperatur ist. Emissionsgrade einiger Baustoffe sind in Tabelle 2.3 gegeben, für Materialien bei denen ein Temperaturbereich angegeben ist, darf für den Emissionsgrad linear interpoliert werden.

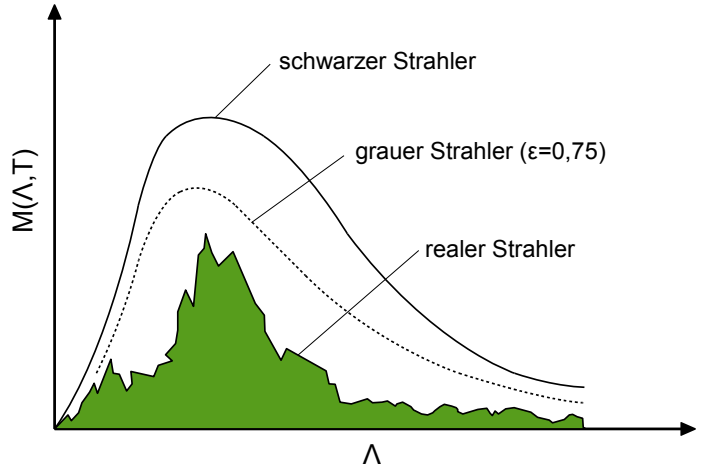


Abbildung 2.8: *Spektrale spezifische Ausstrahlung verschiedener Strahler*

Das Modell des grauen Lambert-Strahlers mit $\alpha(T) = \varepsilon(T)$ trifft bei der Betrachtung von Solarstrahlung (die ihr Maximum im kurzwelligen Bereich hat) nicht zu. Der reale Emissionsgrad der meisten Materialien nimmt bei kleinen Wellenlängen ($< 2\mu m$) deutlich andere Werte an als bei großen Wellenlängen. Zur Berechnung von Solarstrahlung hat sich ein solarer Absorptionsgrad α_S etabliert, der beim Auftreffen von Sonnenstrahlung auf eine Oberfläche anstelle des Absorptionsgrad ($\alpha(T)$) verwendet wird. Der solare Absorptionsgrad wurde für viele Materialien anhand von Messungen ermittelt und kann u.a. [79] entnommen werden.

Die meisten Körper absorbieren auftreffende Wärmestrahlung bereits in einer sehr dünnen oberflächennahen Schicht. Neben den Fluiden stellen Festkörper wie Glas eine Ausnahme dar, die die Wärmestrahlung in einem meist kurzwelligen Wellenlängenbreich durchlassen. Dieser Effekt lässt sich durch den sogenannten Transmissionsgrad $\tau(\Lambda, T)$ beschreiben, der den vom Körper durchgelassenen Energieanteil in Abhängigkeit der Wellenlänge und der Temperatur angibt. Aus der Energiebilanz am Körper (siehe hierzu auch Abbildung 2.2) ergibt sich der Zusammenhang zwischen Reflexionsgrad $\rho(\Lambda, T)$, Transmissionsgrad $\tau(\Lambda, T)$ und Absorptionsgrad $\alpha(\Lambda, T)$ mit:

$$\rho(\Lambda, T) + \tau(\Lambda, T) + \alpha(\Lambda, T) = 1. \tag{2.21}$$

Ohne Berücksichtigung der wellenlängenabhängigen Transmission können Prozesse wie der Treibhauseffekt nicht berechnet werden. Hierbei wird kurzwellige Sonnenstrahlung durch Glasfenster durchgelassen, die dann auf Bauteile im inneren auftreffen und diese erwärmen. Die so aufgeheizten Bauteile emittieren langwellige Strahlung, welche von den Glasfenstern absorbiert und nicht durchgelassen wird. Dieser Effekt führt zu einer kontinuierlichen Erwärmung des Innenraums. Transmissionsgrade realer Körper in Abhängigkeit der Wellenlängen und der Dicke des Körpers sind u.a. in der DIN 1349 [32] aufgeführt.

Zusammenfassend kann festgehalten werden, dass Lambert Strahler eine brauchbare Näherung für reale Körper darstellen und zu erheblichen Vereinfachungen der Strahlungsbe-

Stoff	Temperatur $T°C$	Gesamtemissionsgrad in Richtung der Flächennormale ε_n	Hemisphärischer Gesamtemissionsgrad ε
Buchenholz	70	0,94	0,91
Emaillelack, weiß	20	0,91	
Papier, weiß, matt	95	0,92	0,89
Sand	20	0,76	0,91
Wasser	10...50	0,965	0,91
Eisen, poliert	-73...727	0,32...0,60	0,06...0,25
Eisen, oxidiert	-73...727	0,32...0,60	
Eisen, blank	25	0,24	0,06...0,25
Eisen, angerostet	25	0,61	
Gold, poliert	227...627	0,020...0,035	
Gold, oxidiert	-173...827		0,013...0,070
Kupfer, poliert	327...727	0,012...0,019	
Kupfer, oxidiert	130	0,76	0,725
Aluminium			0,04
Platin			0,05

Tabelle 2.3: *Beispiele für Emissionsgrade einiger Baustoffe (aus [9])*

rechnung führen. Die Annahme grauer Oberflächen trifft jedoch nicht in jedem Fall zu (insbesondere bei der Berechnung von Solarstrahlung) und muss in Abhängigkeit vom Anwendungsfall beurteilt werden. Allerdings ergeben sich bereits bei den zur Verfügung stehenden Materialkennwerten Einschränkungen, da nur für wenige Stoffe umfassende experimentelle Messungen durchgeführt wurden. Die vorhandenen Ergebnisse beschränken sich häufig auf wenige ausgewählte Wellenlängen und Winkelabhängigkeiten nur in Richtung der Flächennormalen. Auch ergeben sich starke Schwankungen in den durchgeführten Messungen, da die Materialeigenschaften der Oberfläche stark von ihrer Beschaffenheit abhängen. Bereits leichte Verunreinigungen oder Oxidbildung sowie unterschiedliche Rauigkeiten können die Eigenschaften des Materials völlig verändern [9].

2.3 Wärmeleitung

Wärmeleitung (heat conduction), auch als Wärmediffusion bezeichnet, ist der Wärmefluss in einem Festkörper oder ruhenden Fluid aufgrund eines vorhandenen Temperaturgradienten [9, 95]. Der Energietransport erfolgt hierbei zwischen benachbarten Molekülen und ist nicht mit dem Transport von Teilchen verknüpft. In Festkörpern erfolgt der Transport von Energie ausschließlich durch Wärmeleitung, in Fluiden wird die Wärmeleitung durch Strö-

mung (Konvektion) und durch Wärmestrahlung überlagert. Der Energietransport in wärmeleitenden Materialien wird durch das Vektorfeld der Wärmestromdichte beschrieben:

$$\dot{q} = \dot{q}(x,t). \tag{2.22}$$

Dieses beschreibt die Wärmestromdichte an einem Ort x in Abhängigkeit der Zeit t. Im Folgenden soll die phänomenologische Erfassung der Wärmeleitung erläutert werden. Eine detaillierte Beschreibung des diffusiven Wärmetransports kann Baehr [9], Boeckh [11], Polifke [95] oder Welty [122].

entnommen werden.

2.3.1 Das Fouriersche Gesetz der Wärmeleitung

Bereits im Jahr 1822 erkannte der Mathematiker und Physiker Jean Baptiste Joseph Fourier die grundlegenden Zusammenhänge der Wärmeleitung. Er konnte aufgrund von experimentellen Beobachtungen die Beziehungen zwischen Wärmefluss $\dot{q}$ und Temperatur T bestimmen. Das nach ihm benannte Fouriersche Gesetz der Wärmeleitung (Gleichung 2.23) beschreibt den Wärmefluss aufgrund eines Temperaturgradienten $\nabla T(\vec{x})$ und der stoffspezifischen Wärmeleitfähigkeit λ (vgl. Abbildung 2.9).

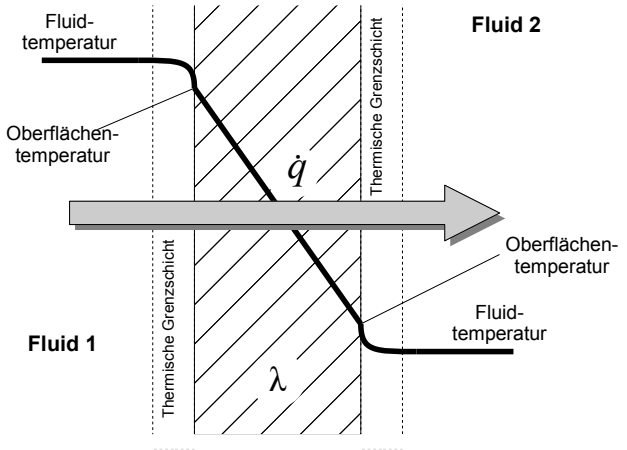


Abbildung 2.9: *Wärmeleitung durch eine Wand*

Durch das negative Vorzeichen wird berücksichtigt, dass der Vektor der Wärmestromdichte stets in Richtung abnehmender Temperatur weist.

$$\vec{\dot{q}}(\vec{x}) = -\lambda \nabla T(\vec{x}) \tag{2.23}$$

Die Wärmeleitfähigkeit λ ist eine Materialeigenschaft in Abhängigkeit der Temperatur und im geringen Maße des Drucks. Nicht selten können für Gase und Metalle im Bereich 100K

bis 1000K Änderungen um mehr als eine Größenordnung beobachtet werden. Für praktische Zwecke können jedoch für geringe Gradienten konstante Werte angenommen werden. In Tabelle 2.4 ist die Wärmeleitfähigkeit bei 20°C und die Rohdichte für einige Baustoffe dargestellt.

Stoff	ϱ in $[kg/m^3]$	λ in $[W/(mK)]$	c in $[J/(kgK)]$
Aluminium	2710	221,0	896
Kupfer	8920 - 8960	350,0 - 399,0	381
Legierte Stähle	7900	15,0 - 42,0	470
Eisen	7700	81,0	439
Mauerwerk (Lochziegel)	600 - 650	0,5 - 1,4	840
Beton	1800 - 2450	2,1	880
Holz (senkrecht zur Faser)	400 - 800	0,09 - 0,19	170
Schaumstoffplatten	25 - 40	0,035 - 0,05	1200
Bitumen	1000	0,16	920
Granit	2750	2,9	790
Wasser	999,975	0,58	3330
Luft	1,204	0,0261	1005

Tabelle 2.4: *Wärmeleitfähigkeit λ und Rohdichte ϱ und spezifische Wärmekapazität c verschiedener Materialien bei 20°C (aus [9])*

Es hat sich herausgestellt, dass das Fouriersche Gesetz (Gleichung 2.23), für verschiedene Materialien und sehr unterschiedlich große Temperaturgradienten, das Wärmeleitungsverhalten sehr genau beschreibt. Zur Berechnung des räumlich und zeitlich abhängigen Temperaturfeldes $T(x,t)$ ist der Fouriersche Ansatz nicht ausreichend. Die Herleitung der vollständigen Wärmeleitungsgleichung wird im folgenden Abschnitt erläutert.

2.3.2 Die Fouriersche Differentialgleichung für das Temperaturfeld

Mit dem Fourierschen Gesetz der Wärmeleitung (Gleichung 2.23) lässt sich für ein Temperaturfeld die zugehörige Wärmestromdichte in einem wärmeleitenden Körper oder Fluid bestimmen. Die zeitliche Abhängigkeit des Temperaturfeldes erhält man, basierend auf dem Energieerhaltungssatz, durch Aufstellen der Leistungsbilanz an einem infinitesimalen Volumenelement eines wärmeleitenden Körpers [9]. Hierbei besitzt nach dem ersten Hauptsatz der Thermodynamik jedes System eine innere Energie U, die sich nur aufgrund eines Wärmestroms $\dot{Q}$ und einer Leistung P ändert:

$$\frac{dU}{dt} = \dot{Q}(t) + P(t). \tag{2.24}$$

Die zugeführte Leistung P setzt sich aus der Volumenleistung P_V und der im Inneren des Systems dissipierten Leistung P_{diss} zusammen. Für inkompressible Körper, die unter Druckeinwirkung keine Volumenänderung aufzeigen, ist $P_V = 0$. Das Stoffmodell des inkompressiblen Körpers stellt für die Wärmeleitung in Festkörpern eine gute Näherung dar. Die dissipierte Leistung P_{diss} ist die von außen zugeführte elektrische Leistung und hängt von der volumenbezogenen Leistungsdichte $\dot{W}$ ab:

$$P(t) = P_{diss}(t) = \int_V \dot{W}(T,x,t)\, dV. \tag{2.25}$$

Die innere Energie U beschreibt das Speichervermögen eines Körpers in Abhängigkeit seiner spezifischen Wärmekapazität c und seiner Dichte ϱ (für inkompressible Körper mit konstanter Dichte ϱ):

$$\frac{dU}{dt} = \varrho \int_V c(T) \frac{\partial T}{\partial t}\, dV. \tag{2.26}$$

Die hier auftretende Stoffeigenschaft c (spezifische Wärmekapazität) beschreibt die Wärmemenge, die einem Stoff pro Kilogramm zugeführt werden muss, um seine Temperatur um ein Grad Kelvin zu erhöhen. Hierbei stehen Masse m, Temperturdifferenz ΔT und die zugeführte Wärmemenge ΔQ in folgendem Zusammenhang:

$$c = \frac{\Delta Q}{m \Delta T} \quad \left[\frac{J}{kg\ K}\right]. \tag{2.27}$$

Für einige typische Materialien ist die spezifische Wärmekapazität in Tabelle 2.4 aufgeführt. Zur Berechnung des Wärmestroms $\dot{Q}$ werden alle über die Oberflächen hineinfließenden Wärmeströme eines Elements betrachtet. Somit folgt für $\dot{Q}$ folgender Ansatz:

$$\dot{Q} = -\int_A \dot{q} n\, dA = -\int_V div\ \dot{q}\, dV, \tag{2.28}$$

mit dem Fluss in Normalenrichtung, bei dem das Oberflächenintegral nach Gaußschem Integralsatz in ein Volumenintegral der Divergenz von $\dot{q}$ umgewandelt wird. Durch Einsetzen von Gleichung 2.25, Gleichung 2.26 und Gleichung 2.28 in den ersten Hauptsatz der Thermodynamik (Gleichung 2.24) erhält man die folgende Gleichung:

$$\varrho \int_V c(T) \frac{\partial T}{\partial t}\, dV = -\int_V div\ \dot{q}\, dV + \int_V \dot{W}(T,x,t)\, dV. \tag{2.29}$$

Durch Einsetzen des Fourierschen Gesetz (Gleichung 2.23) folgt für ein infinitesimales Volumenelement die allgemeine Wärmeleitungsgleichung:

$$\varrho c(T) \frac{\partial T}{\partial t} = div[\lambda(T)\ \nabla T(\vec{x})] + \dot{W}(T,x,t). \tag{2.30}$$

Eine mathematisch geschlossene Form der Wärmeleitungsgleichung erhält man für konstante Stoffwerte. Werden die Temperaturabhängigkeit der Wärmeleitfähigkeit λ und der spezifischen Wärmekapazität c vernachlässigt, nimmt Gleichung 2.30 die folgende Form an:

$$\frac{\partial T}{\partial t} = a\nabla^2 T + \frac{\dot{W}}{c\varrho}, \tag{2.31}$$

mit der Temperaturleitfähigkeit

$$a = \frac{\lambda}{c\varrho}. \tag{2.32}$$

Für kartesische Koordinaten folgt hieraus:

$$\boxed{\frac{\partial T}{\partial t} = a\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2}\right) + \frac{\dot{W}(x,y,z,t,T)}{\varrho\, c}.} \tag{2.33}$$

2.3.3 Zeitliche und örtliche Randbedingungen

Mit der im vorherigen Abschnitt hergeleiteten Wärmeleitungsgleichung lassen sich Temperaturfelder nur im Inneren von Körpern beschreiben. Zur vollständigen Lösung der Differentialgleichung müssen Grenzbedingungen des Körpers berücksichtigt werden. Diese werden durch zeitliche Anfangsbedingungen und Randbedingungen an den Oberflächen des Körpers bestimmt.

Für instationäre Temperaturverteilungen wird eine Anfangsbedingung zur Zeit $t = 0$ benötigt. Diese schreibt eine Temperatur zu einem bestimmten Zeitpunkt für das örtliche Temperaturfeld vor:

$$T(x,y,z,t=0) = T_0(x,y,z). \tag{2.34}$$

Die örtlichen Randbedingungen lassen sich in drei Arten einteilen:

- Dirichlet-Randbedingung (Randbedingung der ersten Art),
- Neumann-Randbedingung (Randbedingung der zweiten Art),
- Randbedingung der dritten Art.

Dirichlet-Randbedingung

Die Dirichlet-Randbedingung (Randbedingung der ersten Art) wird zur Vorgabe von gegebenen Oberflächentemperaturen T_W („Wandtemperaturen“) verwendet (vgl. Abbildung 2.10). Diese werden als Funktion der Temperatur in Abhängigkeit der Zeit und des Ortes vorgegeben, hier am Beispiel des linken Randes für $x = 0$:

$$T_W = T(x=0,t) = f(t). \tag{2.35}$$

Randbedingung der 1. Art sind mathematisch am einfachsten zu modellieren, insbesondere für zeitunabhängige konstante Oberflächentemperaturen.

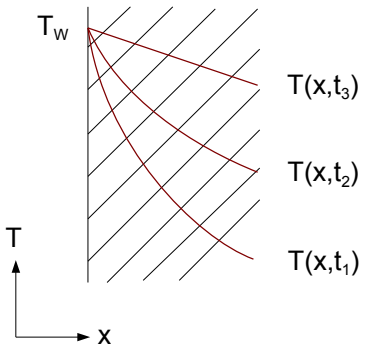


Abbildung 2.10: *Dirichlet Randbedingung: Temperaturverlauf innerhalb eines Körpers aufgrund einer vorgegebenen Oberflächentemperatur T_W*

Neumann-Randbedingung

Die Neumann-Randbedingung (Randbedingung der zweiten Art) wird zur Vorgabe von Wärmeflüssen $\dot{q}_W$ („Wandwärmefluss“) an der Körperoberfläche verwendet (vgl. Abbildung 2.11). Dabei wird basierend auf dem Fourierschen Gesetz (Gleichung 2.23) ein Wert auf dem Rand für die Normalableitung der Lösung vorgegeben. Der Temperaturgradient ist hierbei immer senkrecht zur Oberfläche. Die Wärmestromdichte $\dot{q}_W$ am Rand lässt sich durch den folgenden Ausdruck beschreiben:

$$\dot{q}_W = -\lambda \frac{\partial T}{\partial n}. \tag{2.36}$$

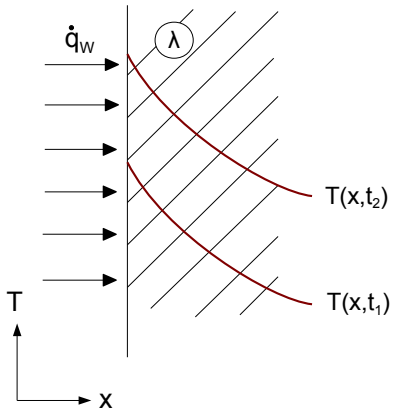


Abbildung 2.11: *Neumann-Randbedingung: Temperaturverlauf innerhalb eines Körpers aufgrund einer vorgegebenen Wärmestromdichte $\dot{q}_W$*

Randbedingungen der dritten Art

Randbedingungen der dritten Art beschreiben den Wärmeübergang zwischen einem Festkörper und einem strömenden Fluid [9]. Der Wärmetransport in einem Fluid erfolgt neben

der Wärmeleitung durch eine makroskopische Bewegung des Fluids. Dieser Vorgang wird als Konvektion bezeichnet, bei der man zwischen zwischen freier und erzwungener Konvektion unterscheidet. Bei der freien oder natürlichen Konvektion wird der Teilchentransport ausschließlich durch Temperaturunterschiede ausgelöst (z.B. Auf- und Abtrieb infolge eines Dichteunterschiedes ausgelöst durch eine Temperaturänderung). Die erzwungene Konvektion erfolgt durch eine äußere Einwirkung auf das Fluid (z.B. Pumpen oder Gebläse).

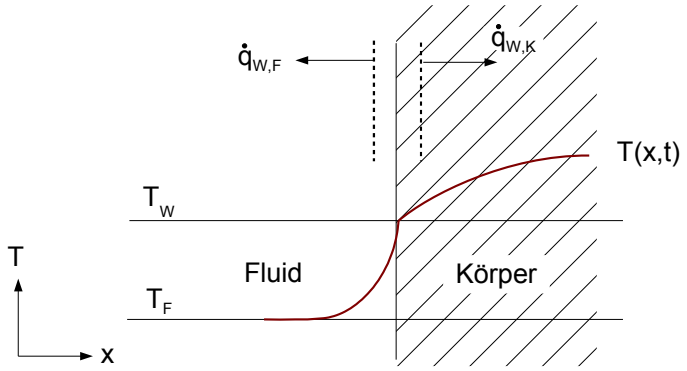


Abbildung 2.12: *Randbedingungen der dritten Art: Konvektiver Wärmeübergang*

Für den Wärmeübergang zwischen einem strömenden Fluid und einem Festkörper ist die dünne Fluidschicht in Wandnähe relevant. Diese wird auch als thermische Grenzschicht bezeichnet, in der die lokale Strömungsgeschwindigkeit sehr gering ist, so dass eine lineare Wärmeleitung vorliegt, welche kontinuierlich durch Mischungsvorgänge überlagert wird und in einen nichtlinearen Temperaturverlauf übergeht (vgl. Abbildung 2.9 und Abbildung 2.12). Die Temperatur in der Grenzschicht ändert sich von der Oberflächentemperatur T_W zu dem Wert T_F (Umgebungstemperatur) in einigem Abstand von der Oberfläche. Aufgrund des auftretenden Temperaturunterschieds $T_W - T_F$ stellt sich eine Wärmestromdichte $\dot{q}_W$ ein, die in komplizierter Weise von der Temperatur- und Geschwindigkeitsfeld im Fluid abhängt. Zur leichteren Betrachtung dieser Abhängigkeit wurde eine spezifische Kennzahl, der Wärmeübergangskoeffizient α_{WF}, eingeführt [9, 95]. Dieser beschreibt die Fähigkeit eines Fluids, thermische Energie zwischen dem Fluid und einem Festkörper auszutauschen und hängt neben Wärmeleitfähigkeit, Dichte und der spezifischen Wärmekapazität auch von der Art und Geschwindigkeit der Strömung ab. Der lokale Wärmeübergangskoeffizient ist definiert als:

$$\alpha_{WF} = \frac{\dot{q}_W}{T_W - T_F}. \tag{2.37}$$

Für die Wärmestromdichte $\dot{q}_W$ zwischen Wand und Fluid gilt somit:

$$\dot{q}_W = \alpha_{WF}(T_W - T_F). \tag{2.38}$$

Da in unmittelbarer Wandnähe die Strömungsgeschwindigkeit gegen null geht, wird die Energie hier nur durch Wärmeleitungsprozesse transportiert. Daraus folgt nach dem Fourierschen Gesetz der Wärmeleitung (vgl. Gleichung 2.23) für die Wärmestromdichte:

$$\dot{q}_W = -\lambda \frac{\partial T}{\partial n}. \tag{2.39}$$

Stellt man nun die Energiebilanz für den wandnahen Wärmefluss im Fluid und den Wärmefluss an der Innenseite des Körpers auf, folgt die Definition der Randbedingung der dritten Art:

$$\alpha_{WF}(T_W - T_F) = -\lambda \frac{\partial T}{\partial n}. \tag{2.40}$$

Hieraus ergibt sich der Wärmeübergangskoeffizient aus der Steigung des Temperaturprofils an der Oberfläche sowie durch den Gradienten aus Oberflächen- und Fluidtemperatur:

$$\alpha_{WF} = -\lambda \frac{\frac{\partial T}{\partial n}}{T_W - T_F}. \tag{2.41}$$

Für die Berechnung des Wärmeübergangskoeffizienten wird das Temperaturfeld im Fluid benötigt, welches durch die Strömungsgeschwindigkeit beeinflusst wird. Somit ist für eine vollständige Betrachtung des Problems die Lösung des Strömungsfeldes notwendig. Das physikalische Verhalten von Fluiden wird im Rahmen dieser Arbeit nicht weiter behandelt, für eine detaillierte Darstellung sei auf Johnson [62] und Kuhlmann [71] verwiesen.

Kopplungsbedingung

Eine weitere Form von Randbedingung ist die Kopplungsbedingung die bei aneinandergrenzenden Festkörpern oder Fluidschichten auftritt. Hierbei dürfen die Kontaktflächen keine Unstetigkeiten aufweisen. Im einfachen Fall zweier ebener Körper kann die Kopplungsbedingung mit:

$$T_l(x = 0, t) = T_r(x = 0, t) \tag{2.42}$$

und

$$\dot{q}_l(x = 0, t) = \dot{q}_r(x = 0, t) \tag{2.43}$$

ausgedrückt werden. Die Indizes l und r bezeichnen den linken und den rechten Körper, die sich im Punkt $x = 0$ berühren.

2.3.4 Lösungsmethoden der Wärmeleitungsgleichung

Zur Lösung der in Abschnitt 2.3.2 hergeleiteten allgemeinen Wärmeleitungsgleichung (Gleichung 2.33) gibt es verschiedene Ansätze. Betrachtet man stationäre Wärmeleitung, bei der die Temperatur an jeder Stelle des Körpers nicht von der Zeit abhängt und geht außerdem

von geometrisch eindimensionalem Wärmefluss aus, lässt sich für viele Fälle eine analytische (geschlossene) Lösung der Wärmeleitungsgleichung finden. Hierbei hängt der Temperaturverlauf nur von einer räumlichen Koordinate ab. Die Lösung für eindimensionale Geometrien, wie den Wärmefluss durch ebene Platten, Hohlzylinder, Kugeln und Stäbe oder den Temperaturverlauf in Rippen und Nadeln sind in der Literatur vielfach aufgeführt, als Beispiel seien hier Baehr [9], Incropera [61] und Welty [122] genannt. Einfache Anwendungsfälle lassen sich somit berechnen. Komplexere Szenarien mit räumlichen Temperaturfeldern, inneren Wärmequellen und instationärem Verhalten, also zeitabhängigen Temperaturfeldern oder zeitlich veränderlichen Randbedingungen, erfordern kompliziertere mathematische Modelle. Numerische Methoden sind hierbei in vielen Fällen die einzige Möglichkeit zur näherungsweisen Lösung der Wärmeleitungsgleichung.

Zur numerischen Berechnung von Anfangswertproblemen stehen verschiedene Verfahren zur Verfügung, wie die Finite-Elemente-Methode (FEM) oder die Finite-Differenzen-Methode (FDM). Die Finite-Elemente-Methode stellt ein verbreitetes Berechnungsverfahren im Ingenieurwesen dar und hat ihr Hauptanwendungsgebiet in der Strukturmechanik [9, 97]. Die Wärmeleitungsgleichung lässt sich effizient mit FEM lösen. Nachteil des Verfahrens stellt die Verwendung meist impliziter Gleichungslöser dar, welche sich bislang nicht so effizient, wie z.B. die Finite-Differenzen-Methode, hardware-beschleunigt auf Grafikkarten ausführen lassen.

Bei der Finite-Differenzen-Methode werden die Ableitungen der Wärmeleitungsgleichung durch Differenzenquotienten ersetzt. Diese Diskretisierung approximiert die Lösung der Gleichung an diskreten Stellen, die ein Gitternetz in Raum und Zeit bilden. Die Vorteile der FD-Methode stellt die relativ einfache Implementierbarkeit sowie ihre gute Skalierbarkeit auf massiv-parallelen Systemen dar. Aus diesem Grund wurde die FD-Methode im Rahmen dieser Arbeit verwendet. Ein effizienter hardware-beschleunigter Ansatz auf Basis der FD-Methode wird im Abschnitt 4.2 dargestellt.

3 Eine hierarchische Datenstruktur für Gebäudemodelle

Für eine dreidimensionale thermische Gebäudesimulation sind zahlreiche Informationen über das Bauwerk notwendig. Aussagekräftige Simulationen können nur durchgeführt werden wenn eine genaue Beschreibung des realen Gebäudes vorliegt. Hier spielen neben den geometrischen Eigenschaften des Bauwerks auch die Materialeigenschaften der Bauteile, äußere Randbedingungen, wie meteorologische Daten, sowie Informationen über das Nutzungsverhalten der Bewohner eine Rolle (vgl. Abbildung 3.1). Diese Informationen sind in einer übergeordneten Datenstruktur abgelegt, welche die Verwaltung der Daten übernimmt und Methoden für den effizienten Zugriff zur Verfügung stellt. Für die meisten Algorithmen hängt die benötigte Laufzeit und der Speicherplatz massiv von der Verwendung geeigneter Datenstrukturen ab. Somit ist es erst durch eine geschickte Wahl der Datenstrukturen möglich komplexe zeitabhängige Simulationen effizient durchzuführen.

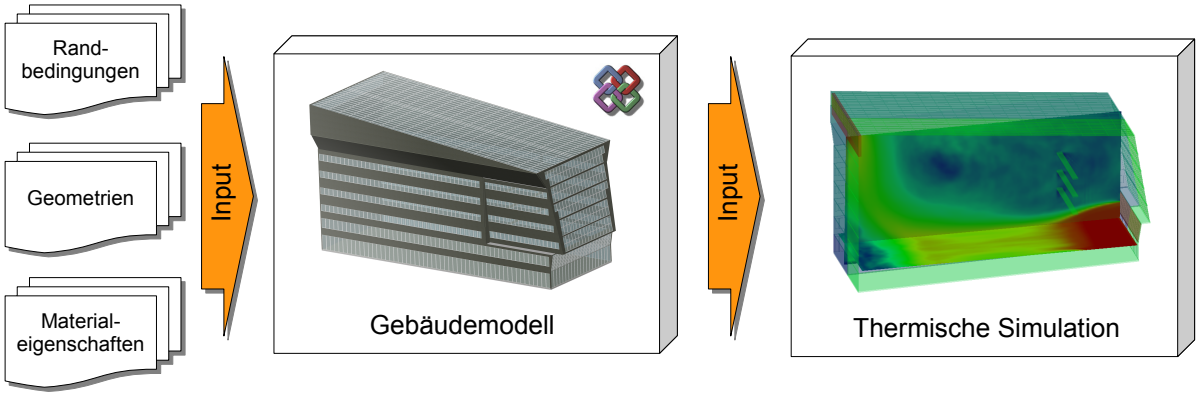


Abbildung 3.1: *Ein übergeordnetes Gebäudemodell für thermische Simulationen*

Vor diesem Hintergrund wurde im Rahmen dieser Arbeit eine effiziente Struktur zur Speicherung von Gebäudemodellen entwickelt, die logische Gebäudestrukturen (z.B. Bauteil-Raum-Gebäude) und zugehörige Eigenschaften abbildet. Als Basis dienen die Industry Foundation Classes (IFC), ein offener Standard zur Beschreibung von Gebäudemodellen, die den Austausch komplexer 3D Planungsdaten zwischen verschiedenen Softwareprodukten im Bauwesen ermöglicht [24] (vgl. Abschnitt 3.1). Somit ist eine Datenaufbereitung und Analyse mit gängigen Programmen wie AutoCAD [7], ALLPLAN [82] oder ArchiCAD [49] möglich. Eine Überblick über die Gebäudedatenmodellierung mit dem IFC-Standard wird in Abschnitt 3.1.1 gegeben.

Die Beschreibung der Geometrie erfolgt durch topologisch verknüpfte Oberflächenmodelle, bei der die Objekte durch ihre begrenzenden Oberflächen beschrieben werden. Die Oberflächenmodelle werden in dafür optimierte Baumstrukturen (Kd-Bäumen, vgl. Abschnitt 3.3) abgelegt, die eine effiziente algorithmische Verarbeitung ermöglichen.

Anmerkung: *Ein **Oberflächenmodell** auch Boundary Representation Model (b-rep) stellt eine Möglichkeit zur Beschreibung von dreidimensionalen Körpern dar. Hierbei wird die Oberfläche des Körpers in eine endliche Anzahl von Teilflächen (häufig einfache geometrische Primitive wie Dreiecke oder Rechtecke) zerlegt. Die Teilflächen werden durch ihre Eckpunkte, Kanten und Flächen beschrieben, die topologisch miteinander verknüpft sind. Oberflächemodelle sind algorithmisch schnell verarbeitbar und werden häufig in der Computergrafik und bei CAD-Anwendungen verwendet.*

Bei der Implementierung der Datenstruktur stand ein strukturierter und wiederverwendbarer Softwareaufbau unter Verwendung objekt-orientierter Programmierparadigmen im Vordergrund. Ein Zugriff auf die Daten ist hierbei im Sinne der Datenkapselung nur über definierte Schnittstellen möglich, wodurch sich die Simulationsalgorithmen leicht austauschen lassen. Beispielsweise wäre es denkbar, die Algorithmik für die thermischen Berechnungen durch Verfahren zur statischen Berechnung des Bauwerks zu ersetzen oder zu ergänzen.

Dieses Kapitel stellt den Aufbau der Datenstruktur zur effizienten Speicherung von Gebäudeinformationen und performant darauf arbeitenden Algorithmen vor. Zunächst wird ein Überblick über die Gebäudedatenmodellierung (Building Information Modeling), insbesondere den im Bauwesen verbreiteten IFC-Standard, gegeben. Anschließend werden der allgemeine Aufbau der Datenstruktur sowie die effiziente Speicherung von Oberflächennetzen mit Hilfe von Kd-Bäumen erläutert. Abschließend wird auf die schnelle Erstellung anisotroper Knotengitter eingegangen.

3.1 Gebäudedatenmodellierung

Die an einem Bauprojekt beteiligten Fachplaner (Tragwerksplaner, Architekten, Haustechniker, Facility-Manager, Bauphysiker, etc.) benötigen verschiedenartige Informationen über das Bauwerk. Für einen Tragwerksplaner sind zum Beispiel die tragenden Bauteile und deren Materialeigenschaften für die Erstellung der erforderlichen Standsicherheitsnachweise von Bedeutung. Architekten interessieren im Rahmen der gestalterischen Planung des Gebäudes Formen, Anordnung und Farbgebung der Bauteile. Dies führt zu mehreren abstrakten Modellen mit unterschiedlicher Sicht auf das Gebäude, die auf die jeweiligen Einsatzgebiete der Planer zugeschnitten sind. Ein Austausch der Modelldaten zwischen den am Bauprojekt beteiligten Unternehmen ist unerlässlich. Da es hierbei häufig zu Schwierigkeiten aufgrund nicht kompatibler Softwareprodukte und Formate kommt, werden Daten redundant vorge-

halten. Dies führt unvermeidlich zu einer inkonsistenten Datenhaltung und höheren Kosten, da Daten mehrfach erfasst werden müssen.

Abbildung 3.2: *Zusammenarbeit an einem Gebäudemodell*

Eine Lösung ist die übergeordnete Zusammenfassung aller Gebäudeinformationen und Partialmodelle innerhalb eines einheitlichen Gesamtmodells (vgl. Abbildung 3.2). Die Integration aller Informationen in einem Modell wird als Produktmodell bezeichnet. Das Produktmodell (ein Begriff aus dem Produktdatenmanagement) beinhaltet alle für die Produktion des Objekts benötigten Daten, zum Beispiel Geometriedaten, Informationen zum Design oder Materialeigenschaften. Zur digitalen Beschreibung von Gebäudemodellen hat sich im Bauwesen der Begriff **Building Information Modeling (BIM)** oder **Gebäudedatenmodellierung** etabliert. Im Sinne des BIM erfolgt die effiziente Planung, Ausführung und Bewirtschaftung von Gebäuden mit Hilfe von Computer-Software. Alle relevanten Gebäudedaten führen hierbei zu einem einheitlichen virtuellen Gebäudemodell. Im Bauwesen hat sich hierzu das IFC-Bauwerksmodell der buildingSMART Organisation als Standard zur Beschreibung solcher Modelle durchgesetzt, welches im folgenden Abschnitt erläutert wird [24].

3.1.1 Das IFC-Bauwerksmodell

Die Industry Foundation Classes (IFC) sind ein offener Standard zur Beschreibung von Gebäudemodellen, der von der buildingSMART (bis September 2009 Industrieallianz für Interoperabilität) definiert wird. Die buildingSMART Organisation gründete sich nach eigenen Angaben mit dem Ziel, „den modellbasierten Ansatz für die Optimierung der Planungs-,

Ausführungs- und Bewirtschaftungsprozesse im Bauwesen im Rahmen der buildingSMART Initiative zu etablieren und dabei die Industry Foundation Classes - IFC - als den offenen Standard für Gebäudemodelle durchzusetzen“ [24]. Die buildingSMART Initiative steht für „den neuen Ansatz, innovative, nachhaltige und kosteneffiziente Gebäude und bauliche Anlagen zu schaffen, indem moderne IT Lösungen mit durchgängiger Datennutzung für integrierte Prozesse genutzt werden“ [24]. Nach eigenen Angaben berücksichtigt die buildingSMART:

- Architekten,
- Ingenieure,
- Vertragsunternehmer,
- Bauherren,
- Facility-Manager,
- Bauunternehmen,
- Softwareentwickler,
- Dienstleister,
- öffentliche Stellen,
- Forschungs-Labors
- und Universitäten.

Hierdurch ergeben sich unterschiedliche Sichtweisen auf ein Gebäudemodell, die in den IFC-Standard einfließen und einen einheitlichen Datenaustausch innerhalb der Bauindustrie ermöglichen sollen.

Der IFC Standard ist unter ISO 16739 registriert und bildet die Grundlage eines Datenmodells zur Beschreibung einer baulichen Anlage. Hierbei wird die Projektstruktur (Grundstück, Gebäude, Gebäudeabschnitte, Geschosse, Raum, Bauteil) auf das Modell übertragen. Dabei verfolgt die IFC einen objektorientierten Ansatz. Hierzu werden Bauteile mit gleichen Eigenschaften zu Klassen zusammengefasst (z.B. Wände, Stützen, Decken, Türen, etc.). Jedes Bauteil enthält neben seinen parametrischen Bauteildaten (wie Länge, Breite, Höhe, etc.) Informationen über seine Beziehungen zu anderen Objekten sowie zugehörige Eigenschaften (Property Sets) und optionale Geometrien (z.B. Oberflächennetze). Die Möglichkeit solche komplexen Informationen über die Bauteile und ihre Beziehungen zwischen Softwareprogrammen austauschen zu können, ist mit dem IFC-Gebäudemodell einzigartig. Jedes erstellte IFC-Objekt erhält eine eindeutige Identifikationsnummer, die eine über die gesamte Lebensdauer des Objekts eindeutige Zuordnung (Identifikation) ermöglicht [58, 59].

Die Definition der Industry Foundation Classes ist an den im Maschinenbau verbreiteten Standard STEP (ISO 10303) zur Beschreibung von Produktdaten angelehnt. Die IFC sind als hierarchisches Schichtenmodell, mit einem von unten nach oben zunehmenden Detaillierungsgrad aufgebaut. Hierdurch lassen sich Teilmodelle auf einen minimalen Kernel aufset-

zen und vom Benutzer an seine speziellen Anforderungen anpassen. Die Struktur besteht aus den vier Schichten (sogenannten Layern) Resource Layer, Core Layer, Interoperability Layer und Domain Layer (vgl. Abbildung 3.3). Die unterste Schicht stellen die Ressourcen dar. Diese definieren allgemeine Konzepte und Klassen, die von den darüberliegenden Schichten verwendet werden, z.B. Material Resource oder Geometric Resource. In der nächsten Ebene folgt der eigentliche Kern des IFC-Modells, der grundlegende Strukturen und Funktionalitäten zur Verfügung stellt. Der dritte Layer definiert Klassen, die von mehreren Bereichen (Domains) benötigt werden. Auf der obersten Ebene folgen die Objekte der einzelnen Anwendungsbereiche. Im aktuellen Release (IFC2x3) der Industry Foundation Classes werden hier neun Anwendungsbereiche (Building Controls, Fire Protection, Structural Elements, Structural Analysis, HVAC, Electrical, Architecture, Construction Management und Facilities Management) abgedeckt.

Softwareprogramme, die den IFC-Standard unterstützen, können Gebäudemodelle in einer neutralen Objektbeschreibung importieren und exportieren, so dass ein Austausch komplexer 3D Planungsdaten zwischen den Planungsbeteiligten möglich wird. Der Austausch erfolgt hierbei durch IFC-Dateien mit der Endung *.ifc. Der IFC-Standard wird von zahlreichen Softwareprogrammen zum Austausch von Gebäudemodellen unterstützt. Anwendungsbereiche sind z.B. Tragwerksplanung, CAD, Architektur und Mengen- und Kostenermittlung. Die aktuelle Spezifikation IFC2x3 erfüllen z.B. die Programme: ALLPLAN von Nemetschek [82], ArchiCAD von Graphisoft [49], AutoCAD Architecture von Autodesk [7], TEKLA Structures 13.0 von TEKLA Corporation [111] und Facility Online von Vizelia [115] um nur einige zu nennen. Die vollständige Liste der IFC zertifizierten Software kann [60] entnommen werden.

Durch die softwareübergreifende Verwendung des IFC-Bauwerksmodells wird ein einheitlicher Datenaustausch von komplexen Gebäudemodellen erst möglich. Hierdurch ergeben sich folgende Vorteile für den Planungsprozess [58]:

- Verbesserte Datenqualität. Durch eine einheitliche Datenbasis werden Inkonsistenzen verringert.
- Größere Flexibilität bei der Auswahl von Fachsoftware.
- Kontinuierliche Verfügbarkeit aller relevanten Daten.
- Geringerer Arbeitsaufwand, da eine Mehrfacherfassung der Daten vermieden wird.
- Bereitstellung von Daten für Nutzer und Behörden.
- Einheitliche Projektdokumentation.
- Verbesserter Informationsaustausch zwischen den Planern.

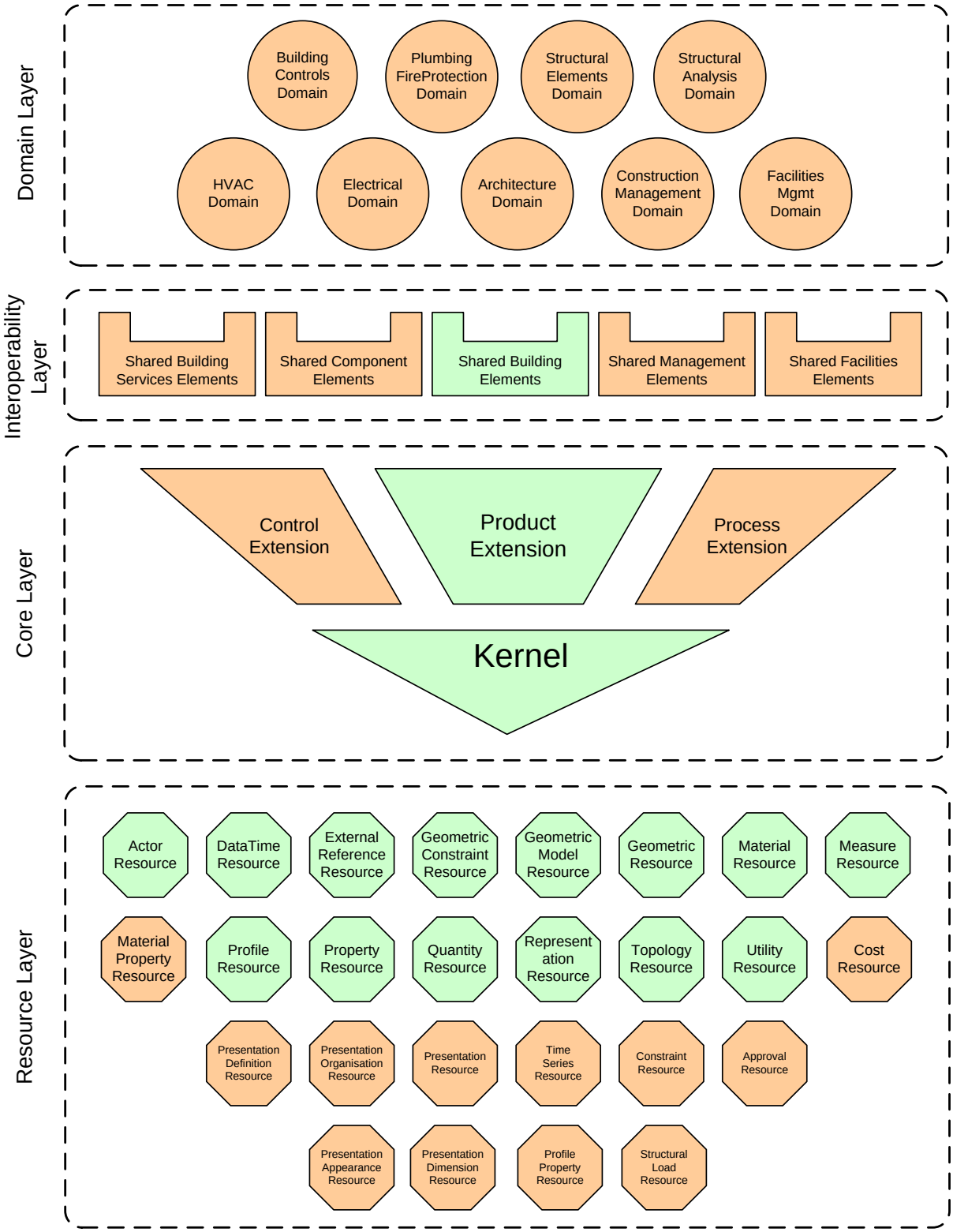


Abbildung 3.3: *IFC-Spezifikation [59]*

3.2 Aufbau der entwickelten Datenstruktur auf Basis des IFC

In diesem Abschnitt wird der Aufbau der im Rahmen dieser Arbeit entwickelten Datenstruktur auf Basis der Industry Foundation Classes erläutert. Die Anforderung an die Datenverwaltung ist es alle Informationen in einem erweiterten Gebäudemodell zu halten. Grundlage stellen hierbei die logischen Gebäudestrukturen dar, welche um benötigte Eigenschaften erweitert sind. Die einzelnen Bestandteile des Bauwerks sind in einer hierarchischen Struktur (Umgebung - Gebäude - Raum - Bauteil) abgelegt und miteinander verknüpft. Ein UML-Klassendiagramm der Datenstruktur ist in Abbildung 3.4 dargestellt.

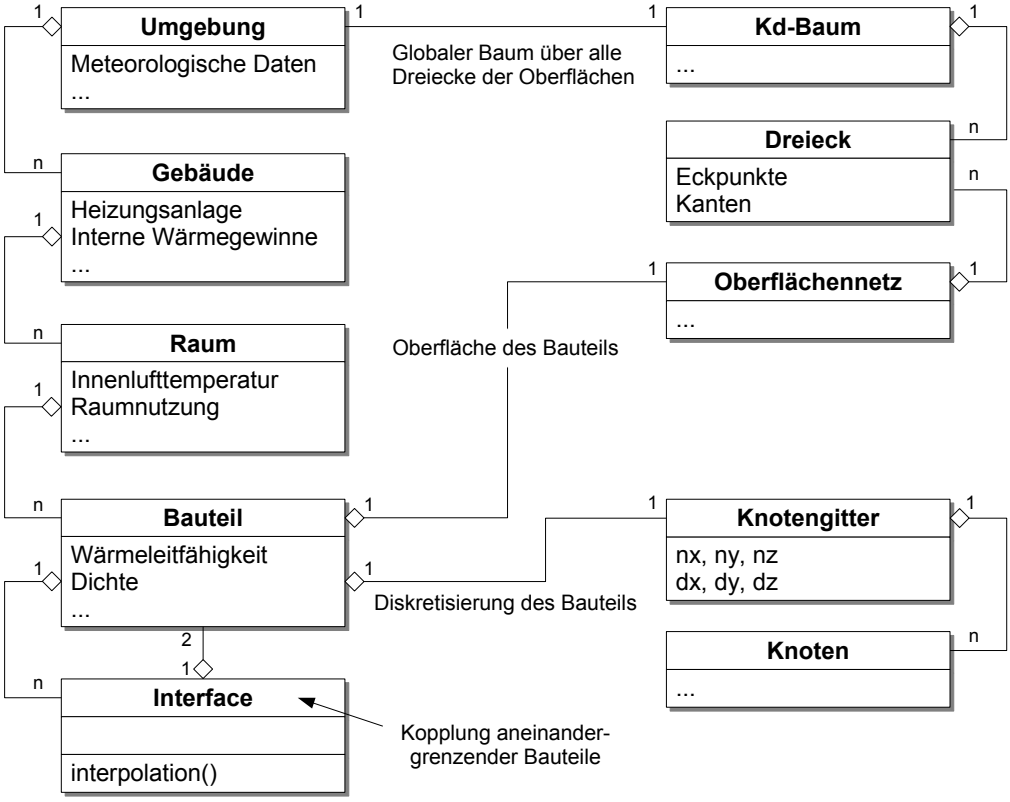


Abbildung 3.4: *UML-Klassendiagramm der Datenstruktur*

Die Umgebung stellt hier die oberste Ebene dar, die neben den meteorologischen Daten eine beliebige Anzahl von Gebäuden enthält. Des Weiteren werden auf dieser Ebene sämtliche Oberflächen der Geometrie in einer Baumstruktur vorgehalten. Diese Struktur auf Basis von optimierten Kd-Bäumen ermöglicht die Ausführung einiger sehr effizienter Algorithmen (z.B. zur Sichtbarkeitsberechnung und für Punkt-in-Polyeder-Tests), die für die thermischen Simulationen benötigt werden. Der Kd-Baum wird hierbei zur Raumzerlegung verwendet und hat sich zur Speicherung unregelmäßig verteilter Flächen komplexer 3D-Umgebungen

etabliert [19, 53, 104, 109]. Eine detailierte Beschreibung des entwickelten Kd-Baums wird in Abschnitt 3.3 gegeben. Auf der nächsten Ebene folgen die Gebäude (erweitert um allgemeine Informationen wie Heizlast oder innere Wärmegewinne). Dem Gebäude untergeordnet sind einzelne Räume mit Daten zur Innenlufttemperatur und der Raumnutzung. Jedem Raum sind seine Bauteile zugeordnet. Die Bauteile sind neben ihren geometrischen Eigenschaften um notwendige Materialeigenschaften erweitert (vgl. Tabelle 3.1) und halten ihr Oberflächenmodell.

Eigenschaft	Bedeutung	Einheit	Datentyp	Standardwert
T	Oberflächentemperatur	$^{\circ}C$	Float	20,0
id	eindeutige Objekt ID	–	Integer	−1
α	Absorptionsgrad	–	Float	0,8
α_S	solarer Absorptionsgrad	–	Float	0,9
ϵ	Emissionsgrad	–	Float	0,8
ρ_d	diffuser Reflexionsgrad	–	Float	0,2
τ	Transmissionsgrad	–	Float	0,0
c	spezifische Wärmekapazität	$J/(kgK)$	Float	0,88
λ	Wärmeleitfähigkeit	$W/(mK)$	Float	2,1
ρ	Dichte	kg/m^3	Float	2400,0

Tabelle 3.1: *Materialeigenschaften der Bauteilobjekte*

Zudem sind die Bauteile elementweise diskretisiert, hierzu werden anisotrope Knotengitter verwendet. Das Knotengitter zerlegt das Bauteil in diskrete Punkte (Knoten) auf denen die Wärmeleitungsgleichung mit der Finite-Differenzen-Methode gelöst wird (vgl. Abschnitt 4.2). Die Struktur des Rechengitters ist von entscheidender Bedeutung für die Effizienz und Genauigkeit des Lösungsverfahrens. Die anisotrope Diskretisierung bietet den Vorteil, dass die Auflösung der Bauteile in die drei Richtungen unterschiedlich gewählt werden kann, um somit auch filigrane Bauteile (wie Fenster) abzubilden. Ein weiterer Vorteil ist, dass in Bereichen, in denen keine hohe Genauigkeit benötigt wird, die Bauteile gröber aufgelöst werden können und somit die Rechenzeit des Verfahrens abnimmt. Die effiziente Erstellung der verwendeten anisotropen Gitter wird in Abschnitt 3.4 erläutert.

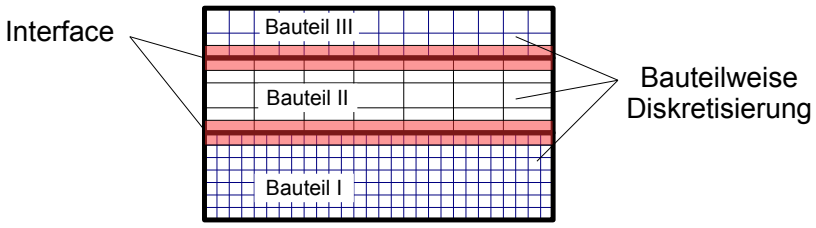


Abbildung 3.5: *Bauteilweise Diskretisierung und Kopplung über Interfaces*

Aneinandergrenzende Bauteile sind über sogenannte Interfaces miteinander verknüpft (vgl. Abbildung 3.5). Stoßen zum Beispiel zwei Wände aneinander, so findet ein Wärmefluss über ihre Kontaktfläche statt (vgl. Abbildung 3.5). Für diesen Wärmefluss sind aufgrund der unterschiedlichen Diskretisierung der Bauteile räumliche Interpolationen notwendig. Der Wärmeaustausch (und die benötigten Interpolationen) zwischen den Bauteilen werden über die Interfaces realisiert (vgl. Abbildung 3.5). Ein Interface verbindet hierbei immer zwei Bauteile. Eine ausführliche Beschreibung der entwickelten Methoden erfolgt in Abschnitt 4.2.3.

Für den Im- und Export von IFC-Gebäudemodellen wurde eine externe Bibliothek, die Open IFC-Toolbox [114], verwendet. Diese objekt-orientierte Schnittstelle ermöglicht das Lesen und Schreiben von IFC-Dateien und ist zum aktuellen Standard IFC2x3 kompatibel. Somit ist eine softwareübergreifende Verwendung des Bauwerksmodells durch einen einheitlichen Datenaustausch möglich. Alle für thermische Simulationen relevanten Gebäudedaten sind in das virtuelle Gebäudemodell integriert. Die Aufbereitung und Optimierung von Geometrien, Randbedingungen und weiteren Metainformationen ist somit mit Standardsoftware möglich. In Abbildung 3.6 ist der Ausschnitt eines exportierten IFC-Gebäudemodells eines Großraumbüros dargestellt, das mit dem IfcViewer [63] sowie mit AutoCAD [7] eingelesen wurde.

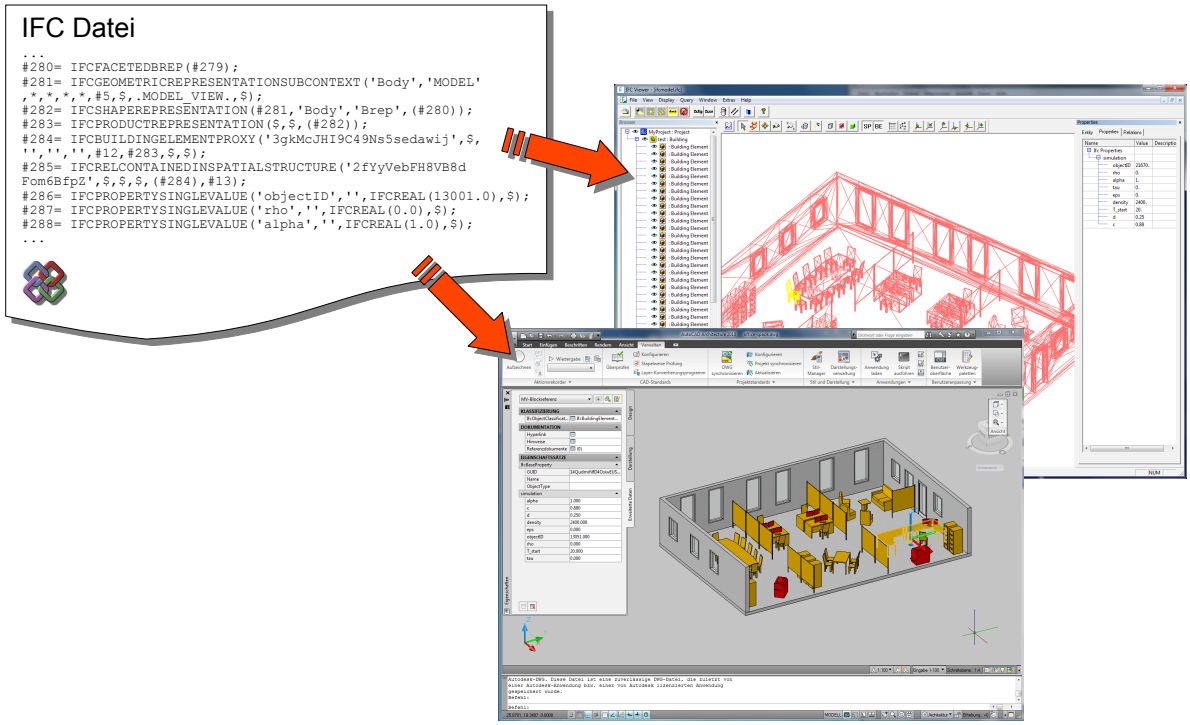


Abbildung 3.6: *Darstellung des exportierten Gebäudemodells in AutoCAD und dem IfcViewer*

3.3 Kd-Bäume zur Speicherung von Oberflächennetzen

Der Raytracing Algorithmus (dt. Strahlenverfolgung) stellt die fundamentale Operation bei der Berechnung des Strahlungsaustausches (vgl. Abschnitt 4.1), sowie bei der Erstellung anisotroper Gitter zur Berechnung der Wärmeleitung (siehe Abschnitt 3.4) dar. Die Beschleunigung des Raytracing Prozesses ist somit ein wichtiger Faktor zur Entwicklung eines performanten Simulationskerns. Raytracing kann effizient unter Verwendung so genannter Raumpartitionierungsverfahren (space partitioning) erfolgen. Diese Verfahren waren in den letzten Jahrzehnten Gegenstand intensiver Forschung. Hierbei konnten sich Kd-Bäume gegenüber anderen Verfahren wie reguläre Gitter [42], Octrees [45] und Bounding-Volume-Hierarchien (BVH) als Standardverfahren etablieren [53, 104, 109]. Die Komplexität des Raytracing Algorithmus auf Kd-Bäumen reduziert sich damit, zwischen zwei Objekten und n im Raum vorhandenen Objekten, auf $O(log\ n)$ bei linearem Speicherverbrauch.

Ein k-dimensionaler Baum (Kd-Baum) ist eine besondere Form eines unbalancierten binären Suchbaums, vorgestellt von Bentley im Jahr 1975 [13].

> **Anmerkung:** *Ein **Baum** ist eine in der Informatik häufig eingesetzte hierarchische Datenstruktur, bei der Elemente (die sogenannten Knoten) einer Menge, über Kanten miteinander verbunden sind. Hierbei ist jeder Knoten, ausgehend von dem Wurzelknoten (von dem aus alle anderen Knoten erreichbar sind) über genau einen Weg zu erreichen. Bei einem Binärbaum folgen auf jeden Knoten zwei Kindknoten; als Kindknoten bezeichnet man einen direkten Nachfolger. Die Tiefe eines Knotens ist als die Anzahl der Kanten im Weg von der Wurzel bis zum Knoten definiert.*

Ursprunglich wurde er zur effizienten Verwaltung von mehrdimensionalen Datenpunkten verwendet und ermöglicht eine schnelle assoziative Suche. Des Weiteren haben sich Kd-Bäume für das Raytracing als sehr effiziente Variante zur binären Raumpartitionierung (Binary Space Partitioning (BSP)) von Oberflächennetzen erwiesen, bei welcher der Raum durch achsenparallele Hyperebenen in Quader unterteilt wird. Der Kd-Baum wird hierbei über die umschließenden, an den Achsen ausgerichteten Quadern (Axis-Aligned Bounding Boxes (AABB)) der einzelnen Netzelemente aufgebaut (z.B. Dreiecke oder Rechtecke). Diese Oberflächenpolygone sind meistens Dreiecke oder Rechtecke und werden als Patches bezeichnet. Die Wurzel des Baums enthält alle Patches der Umgebung. In jedem Schritt der rekursiven Konstruktion wird nun der aktuelle Knoten durch eine achsenparallele Ebene in zwei Kindknoten unterteilt. Diesen werden dann den innerhalb des Knotens liegenden Patches zugeordnet (vgl. Abbildung 3.7). Dieser Prozess wird wiederholt, bis ein Abbruchkriterium erreicht wird (z.B. eine maximale Baumhöhe, eine minimale Anzahl Patches innerhalb der Blätter oder heuristische Ansätze).

Der große Vorteil von Kd-Bäumen gegenüber anderen Raumpartitionierungsverfahren liegt in der verbesserten Anpassungsfähigkeit an die Geometrie und die Maximierung von Berei-

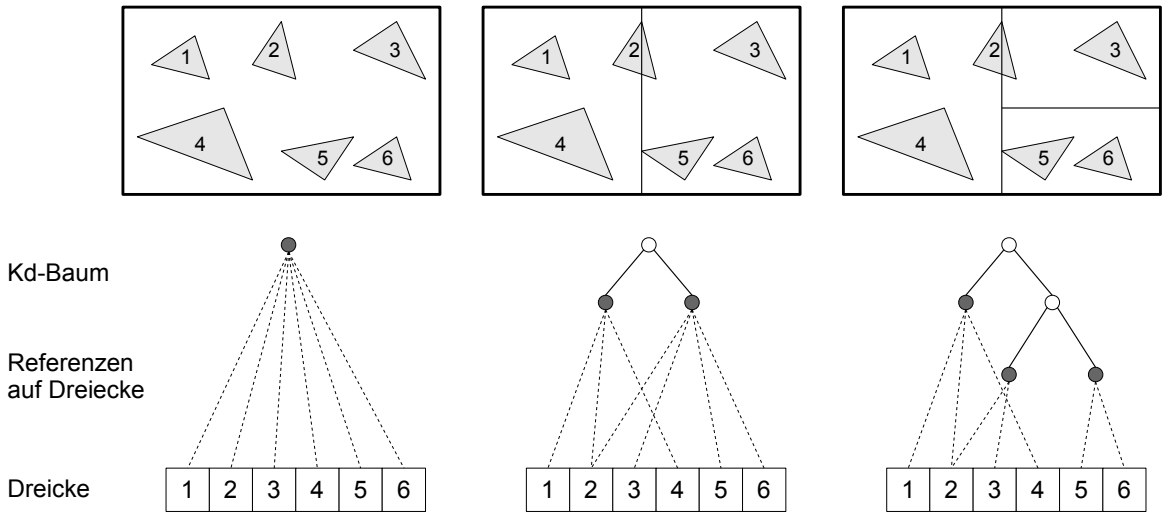
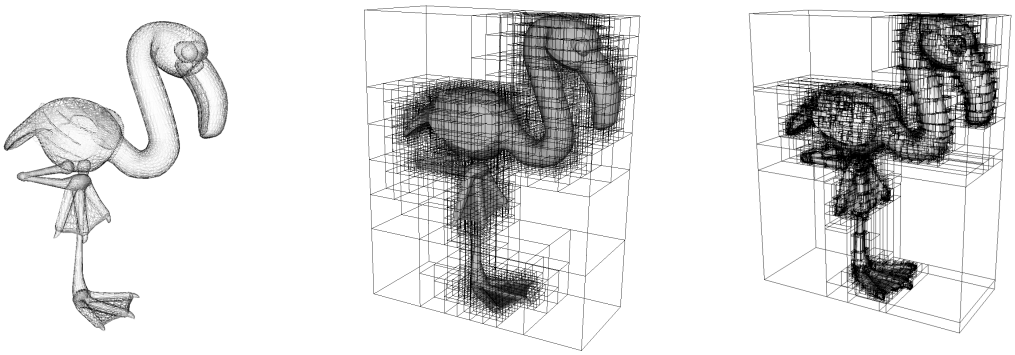


Abbildung 3.7: *Beispiel eines Kd-Baums in 2D*

chen im Baum, in denen keine Geometrie vorhanden ist. Hierbei kann die Qualität signifikant durch die Position der Unterteilungsebene (splitting plane) beeinflusst werden [118]. In den folgenden Abschnitten werden Ansätze für optimierte Kd-Bäume erläutert, welche die Grundlage der in dieser Arbeit entwickelten Datenstruktur darstellen.

3.3.1 Heuristische Verfahren zur Optimierung von Kd-Bäumen

Ein naiver Ansatz zur Unterteilung eines Kd-Baum Knotens ist die Teilung im Median des Knotens. Dieses Verfahren, auch als Spatial Median bezeichnet, lässt sich einfach implementieren und führt zu einer Octree ähnlichen Gebietszerlegung, liefert jedoch nicht die bestmöglichen Ergebnisse.



(a) *Oberflächennetz* ***(b)*** *Kd-Baum (Spatial Median)* ***(c)*** *Kd-Baum (SAH)*

Abbildung 3.8: *Beispiel eines Kd-Baums in 3D*

Andere Methoden verwenden geometrieabhängige Konstanten und führen zu besseren Ergebnissen, allerdings müssen vom Anwender szenenabhängige Einstellungen vorgenommen werden. Um das volle Potential von Kd-Bäumen auszuschöpfen, hat sich ein heuristisches Verfahren, die Surface Area Heuristic (SAH), etabliert [53, 54, 75]. Dieser von MacDonald et al. [75] vorgestellte Ansatz maximiert den Leerraum im Baum durch Minimierung einer heuristischen Kostenfunktion (vgl. Abbildung 3.8).

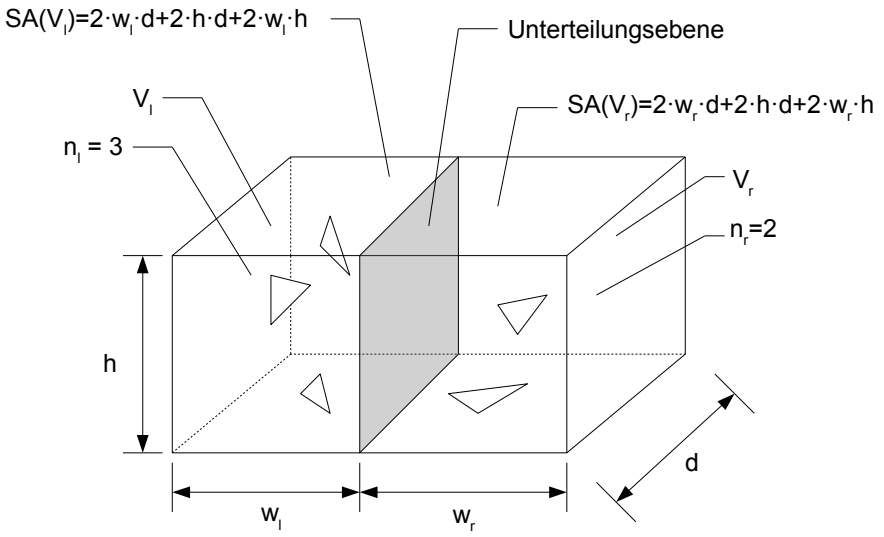


Abbildung 3.9: *Wahl der Unterteilungsebene (splitting plane) in einem Kd-Baumknoten*

Die Kostenfunktion basiert auf der Tatsache, dass die geometrische Wahrscheinlichkeit, dass ein Strahl einen Knoten des Baums schneidet $P(V'|V)$ proportional zur Oberfläche des Knotens $SA(V')$ geteilt durch die Oberfläche des Elternknotens $SA(V)$ ist:

$$P(V'|V) = \frac{SA(V')}{SA(V)}. \tag{3.1}$$

Hierbei wird von folgenden Annahmen ausgegangen: Ursprung und Richtungen der Strahlen sind uniform verteilt, die Kosten eines Traversierungsschritts C_t und die Kosten des Schnittpunkttests zwischen Strahl und Patch C_i sind bekannt. Außerdem sind die Kosten des Schnittpunkttests für n Patches proportional zur Anzahl der Patches. Hieraus lassen sich die Kosten $C_N(V)$ für einen inneren Knoten durch folgende Gleichung berechnen:

$$\begin{aligned} C_N(V) &= C_t + n_l C_i P(V_l|V) + n_r C_i P(V_r|V) \\ &= C_t + C_i \left(n_l \frac{SA(V_l)}{SA(V)} + n_r \frac{SA(V_r)}{SA(V)} \right), \end{aligned} \tag{3.2}$$

mit n_l der Anzahl Patches, die den linken Knoten überschneiden und analog n_r der Anzahl der Patches, die den rechten Knoten überschneiden (vgl. Abbildung 3.9). Die Konstanten C_t (Kosten eines Traversierungsschritts) und C_i (Kosten des Schnittpunkttests zwischen Strahl

und Patch) sind implementierungsabhängig und haben sich mit $C_t = 3.0$ und $C_i = 4.0$ als sinnvoll erwiesen. Die optimale Unterteilungsebene für einen Knoten ist an der Position, welche Gleichung 3.2 minimiert. Somit kann ein optimaler Kd-Baum gefunden werden, bei dem global alle möglichen Unterteilungsebenen berücksichtigt werden. Dieser Ansatz ist jedoch teuer und wenig praktikabel. Hier hat sich ein Ansatz durchgesetzt, bei dem lokal für jede weitere Unterteilung eine optimale Position der Unterteilungsebene gesucht wird (lokale Approximation der Kostenfunktion), ohne dabei Knoten auf anderen Leveln zu berücksichtigen [54, 117].

3.3.2 Konstruktion von Kd-Bäumen

Das Finden der optimalen Unterteilungsebene ist der aufwändigste Schritt beim rekursiven Erstellen des Kd-Baums. Um die Anzahl der zu untersuchenden Unterteilungsebenen zu reduzieren, werden nur Ebenen betrachtet, die am Rand eines Patches liegen, da eine optimale Position immer mit dem Patchrand zusammenfällt. Es ist also ausreichend, die Kostenfunktion an einer endlichen Folge von Stellen (den Rändern der auf die jeweilige Achse projezierten Patches) zu untersuchen [54]. Abbildung 3.10 zeigt die Projektion der Dreiecke auf die x-Achse. Als mögliche Unterteilungsebenen werden nur die Ränder der Dreiecke (a_1 und a_2 von Dreieck A, b_1 und $b_2$von Dreieck B, etc.) betrachtet.

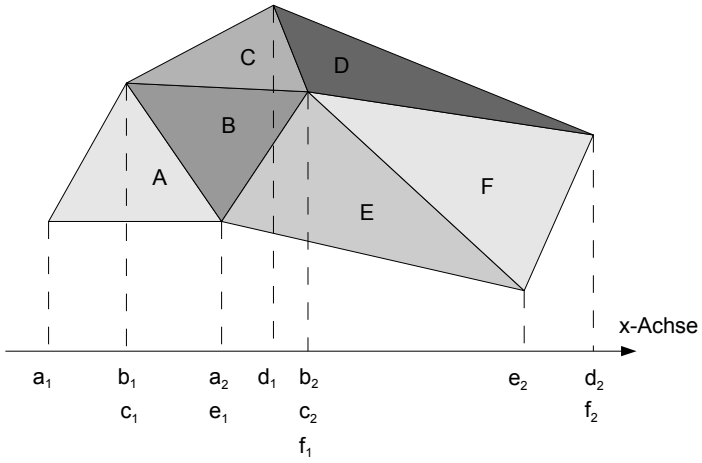


Abbildung 3.10: *Mögliche Unterteilungsebenen an den Rändern der Patches*

Für Dreiecke mit Rändern ganz oder teilweise außerhalb eines Knotens muss besonders verfahren werden. Hierbei wird nur die Überschneidung des Dreiecks mit dem Knoten betrachtet und nicht evtl. außerhalb des Knotens liegende Dreiecksränder. Des Weiteren müssen Sonderfälle wie „flache“ Zellen, bei denen Dreiecksebenen senkrecht zur Unterteilungsachse sind, betrachtet werden. Ein Ansatz ist es, solche flachen Zellen einmal im linken Knoten und einmal im rechten Knoten zu verwenden, um dann die Variante mit den geringeren Kosten zu wählen. Die Berechnung der Kosten mit der Surface Area Heuristic (SAH) unter Berücksichtigung der genannten Sonderfälle ist in Algorithmus 3.1 dargestellt.

Algorithmus 3.1 Surface Area Heuristic (SAH)

function CALCSAH(n_l, n_r , n_p, SA_{VL}, SA_{VR},SA_V) ▷ n_p ist die Anzahl flacher Zellen
  $Cl_{N(V)}$ = CALCCOSTFUNCTION($n_l + n_p$, n_r, SA_{VL}, SA_{VR}, SA_V)
  $Cr_{N(V)}$ = CALCCOSTFUNCTION(n_l, $n_r + n_p$, SA_{VL}, SA_{VR}, SA_V)
  if $Cl_{N(V)} < Cr_{N(V)}$ **then**
    return $Cl_{N(V)}$
  else
    return $Cr_{N(V)}$
  end if
end function

function CALCCOSTFUNCTION(n_l, n_r,SA_{VL}, SA_{VR},SA_V)
  return $C_t + C_i \left(n_l \frac{SA_{VL}}{SA_V} + n_r \frac{SA_{VR}}{SA_V} \right)$
end function

Für jede weitere Unterteilung des Baums müssen somit $6n$ mögliche Unterteilungsebenen untersucht werden (wobei n die Anzahl der Dreiecke ist, die innerhalb des Knotens liegen). Für jede dieser Ebenen muss die Anzahl der im linken Kindknoten liegenden Patches n_l und die Anzahl der im rechten Kindknoten liegenden Patches n_r bestimmt werden. Ein einfacher Ansatz, bei dem für jede mögliche Unterteilungsebene ein weiteres Mal über die Dreiecke iteriert wird um n_l und n_r zu bestimmen, führt zu einer algorithmischen Komplexität von $O(n^2)$. Mit diesem Verfahren ist das schnelle Erstellen des Kd-Baums für komplexe Geometrien nicht möglich. In den letzten Jahren wurden allerdings Algorithmen basierend auf dem Sweep-Verfahren entwickelt, die eine Konstruktion des Baums mit einer Komplexität von $O(n\ log^2\ n)$ erlauben [92, 108].

Anmerkung: *Das **Sweep-Verfahren** beschreibt eine Familie von Algorithmen, bei der eine Sweep-Gerade (in 2D) oder eine Sweep-Ebene (in 3D) durch den Raum bewegt wird. Hierbei werden die von der Sweep-Gerade oder Sweep-Ebene berührten Objekte verarbeitet. Das Verfahren wird häufig bei Problemstellungen der algorithmischen Geometrie (Computational Geometry), wie der Delaunay Triangulation, Schnittpunktbestimmung von Liniensegmenten, etc. angewandt und führt in der Regel zu einer deutlichen Komplexitätreduktion des Problems [14].*

Somit kann der aufwändigste Schritt, die Berechnung von n_l und n_r, effizient gelöst werden. Hierbei werden mit dem Durchlaufen (sweeping) der entlang einer Achse sortierten Unterteilungsebenen, n_l und n_r inkrementell aktualisiert (vgl. Abbildung 3.10). Dieser Ansatz basiert auf der Annahme, dass die Anzahl aller beginnenden und endenden Dreiecke für jeden Eckpunkt eines Dreiecks bekannt sind. Der allgemeine Ablauf des Algorithmus ist im Folgenden dargestellt (vgl. auch Algorithmus 3.2):

1. Bestimmen der möglichen Unterteilungsebenen.
2. Sortieren der Unterteilungsebenen entlang einer Achse.
3. Durchlaufen (sweeping) der Unterteilungsebenen und inkrementelles Aktualisieren von n_l und n_r. Berechnen der Kosten $C_N(V)$ für jede Unterteilungsebene.
4. Wahl der optimalen Unterteilungsebene (mit den minimalen Kosten $C_N(V)$), Prüfen ob das Abbruchkriterium erreicht ist (oder ein weiteres Unterteilen sinnvoll ist) und ggf. Erstellen zwei neuer Kindknoten.
5. Den neuen Knoten die zugehörigen (innerhalb des Knoten liegenden) Dreiecke zuweisen.
6. Rekursive Wiederholung der Prozedur.

Ein verbesserter Ansatz zur Erstellung von Kd-Bäumen wurde von Wald und Havran in [117] publiziert. Hierbei kommt die Konstruktion ohne ein vorheriges Sortieren der Unterteilungsebenen aus, was zu einer Reduktion der Komplexität auf $O(n\ log\ n)$ führt. Eine detaillierte Beschreibung der Algorithmik kann [117] entnommen werden.

Algorithmus 3.2 Suchen der optimalen Unterteilungsebene für einen Knoten V

```
function FINDBESTSPLITCANDIDATE(V)
    minCN = ∞,  bestSplit = null
    for k from 1 to 3 do                          ▷ Schleife über die drei Achsen (x,y,z)
        sc = GETPOSSIBLESPLITCANDIDATES(node, k)
        SORT(sc)
        for for each split candidate s of sc do   ▷ Inkrementelles Durchlaufen (sweeping)
            sp = sc.get(i − 1)                    ▷ Unterteilungsebene die vor s in der Liste steht
            s.nl = sp.nl + sp.starting + sp.np
            s.nr = sp.nr − s.ending + s.np
            C_N(V) = CALCSAH(s.nl, s.nr, s.np, SA_VL, SA_VR, SA_V)
            if C_N(V) < minCN then
                minCN = C_N(V)
                bestSplit = s
            end if
        end for
    end for
    if C_N(V) > n · C_i then                      ▷ Abbruchkriterium überprüfen
        stop subdividing
    else
        return bestSplit
    end if
end function
```

Mit dem vorgestellten heuristischen Ansatz lassen sich optimale Unterteilungsebenen bei der Erstellung von Kd-Bäumen finden. Es stellt sich nun die Frage, bis zu welcher Tiefe der Baum aufgebaut werden sollte, d.h. ab wann ein weiteres Unterteilen nicht mehr sinnvoll ist. Viele Raumpartitionierungsverfahren (z.B. Octrees, Quadtrees, etc.) verwenden Ad-hoc Abbruchkriterien, bei denen die Konstruktion des Baums nach einer maximalen Tiefe der Blätter d_{max} oder einer festgelegten Anzahl Objekte innerhalb der Blätter n_{max} abgebrochen wird. Diese Parameter werden vom Benutzer definiert und haben großen Einfluss auf die Performance des Raytracing Algorithmus. Häufig werden Werte für die maximale Tiefe von $d_{max} = 24$ und die Anzahl Objekte innerhalb der Blätter mit $n_{max} = 4$ festgelegt.

Ein weit besserer Ansatz für die Entscheidung, wie weit der Baum aufgebaut werden soll, wurde von Havran und Bittner in [54] vorgestellt. Sie entwickelten ein automatisches Abbruchkriterium (automatic termination criterion, ATC) basierend auf dem Kostenmodell, welches ohne benutzerspezifische Konstanten auskommt. Hier wird der Unterteilungsprozess beendet, wenn die Kosten der optimalen Unterteilungsebene $C_N(V)_{min}$ teurer sind, als den Knoten nicht weiter zu unterteilen (vgl. Algorithmus 3.2). Zusätzlich wird die Verwendung von d_{max} und n_{max} empfohlen, um Anforderungen an den Speicher zu begrenzen und Effekte bei Patches, die mehrere Knoten überlappen zu vermeiden. Die maximale Baumtiefe sollte von der Anzahl der Patches n der Umgebung abhängen. Havran und Bittner schlagen hier $d_{max} = 1.2\ log_2(n) + 2.0$ vor [54].

3.3.3 Effiziente Traversierung

Die Sichtbarkeitsberechnung zwischen Patches erfolgt, in dem ein Strahl auf den Kd-Baum „geschossen“ wird. Hierbei werden ausgehend vom Wurzelknoten die Kindknoten rekursiv auf Schnittpunkte mit dem Strahl getestet; solange bis ein Blattknoten erreicht ist. Alle Patches innerhalb des Quaders des Knoten werden iterativ auf Schnittpunkte mit dem Strahl getestet (effiziente Algorithmen zur Schnittpunktberechnung von Strahl-Quader und Strahl-Dreieck werden in Abschnitt 3.3.4 gegeben). Durch diesen Ansatz kann die Sichtbarkeitsberechnung zwischen zwei Flächen auf eine Komplexität von $O(log\ n)$ reduziert werden.

In einem Kd-Baum werden Patches häufig von mehreren Blättern referenziert, da sie mehrere Knoten überlappen (vgl. Abbildung 3.7). Das führt dazu, dass ein Strahl mit großer Wahrscheinlichkeit einen Patch mehrfach auf Schnittpunkte testet. Hierdurch kommt es zu verschiedenen Problemen. Zum einen steigt die Laufzeit des Algorithmus an und zum anderen können physikalische Fehler entstehen, insbesondere im Fall teilweise transmittierender Körper, bei denen die Energie eines Strahls durch eine Fläche mehrfach abgemindert wird. Diese Effekte können durch Verwendung des Mailbox-Verfahrens vermieden werden [53, 92, 119]. Für jeden Strahl wird eine Hashtabelle (Mailbox) zur Speicherung der bereits getesteten Patches verwendet. Bevor für einen Patch der Schnittpunkt mit dem Strahl berechnet wird, kann anhand der Hashtabelle geprüft werden, ob der Patch bereits getestet

wurde. Somit können redundante Tests vermieden werden. In Algorithmus 3.3 ist der hierarchische Sichtbarkeitstest zwischen zwei Patches dargestellt.

Algorithmus 3.3 Hierarchischer Sichtbarkeitstest zwischen zwei Patches

```
function VISIBILITY(patch_1, patch_2)
    create new mailbox
    return INTERSECTRAY(patch_1.center, patch_2.center, kd-tree root node, mailbox)
end function

function INTERSECTRAY(p_1, p_2, kdnode, mailbox)
    if bounding box (AABB) of kdnode intersects ray from p_1 to p_2 then
        if kdnode is no leaf node then
            if kdnode has left child c_l then
                INTERSECTRAY(p_1, p_2, c_l, mailbox)              ▷ Rekursiver Aufruf
            end if
            if kdnode has right child c_r then
                INTERSECTRAY(p_1, p_2, c_r, mailbox)              ▷ Rekursiver Aufruf
            end if
        else
            for each patch p inside the kdnode do
                if mailbox not contains p then
                    put p into mailbox
                    if p intersects ray from p_1 to p_2 then
                        return intersection
                    end if
                end if
            end for
        end if
    end if
    return no intersection
end function
```

Ein weiterer Ansatz, der im Rahmen dieser Arbeit nicht weiter verfolgt wurde, ist eine Optimierung des Speicherlayouts der Baumstruktur. Bei der Traversierung des Baums ist das Verhältnis von Berechnung und Speicherzugriff sehr gering. Somit ist es naheliegend, dass durch schlanke Datenstrukturen die Effizienz der Algorithmik gesteigert werden kann. Einige Forschergruppen konnten zeigen, dass speicheroptimierte Datenstrukturen die Laufzeit des Raytracing Algorithmus verringern [116, 119]. Ein Ansatz ist die Speicherung eines Baumknotens in 8 Byte (diese ist durch geschickte Überlagerung der Daten möglich), was zu (je nach Cache Größe) 4 - 16 Knoten pro Cache-Line führt. Eine detaillierte Beschreibung dieser Datenstruktur kann u.a. [116] entnommen werden.

3.3.4 Schnittpunkttest

Beim Raytracing besteht die fundamentale Operation aus der Überprüfung der Schnittpunkte eines Strahls mit Objekten der Umgebung, meistens geometrische Primitive (wie Dreiecke, Quader, etc.). Da diese Operationen vor allem bei komplexen Szenen mit mehreren Millionen Oberflächenelementen sehr häufig durchgeführt werden müssen, sind effiziente Berechnungsalgorithmen von großer Bedeutung. Die beiden im Rahmen dieser Arbeit benötigten Schnittpunkttests Strahl/Quader und Strahl/Dreieck werden in diesem Abschnitt dargestellt. Ein Strahl (eine gerichtete Strecke) ist dabei durch einen Anfangspunkt $o(x_1, y_1, z_1)$ und einen Endpunkt $e(x_2, y_2, z_2)$ festgelegt. Sein Richtungsvektor ist somit durch $d = (x_2 - x_1, y_2 - y_1, z_2 - z_1)$ gegeben.

Neben den thermischen Simulationen tauchen diese Verfahren auch bei der Kollisionserkennung oder der Lichtsimulation auf und sind seit vielen Jahren Gegenstand intensiver Forschung. Ein Überblick über den Stand der Forschung kann z.B. Akenine-Möller und Haines [3] entnommen werden.

Schnittpunkte Strahl/Quader

Ein effizientes Verfahren zur Überprüfung, ob ein Strahl einen Quader (beschrieben durch sechs achsenparallele Flächen AABB) schneidet, ist die sogenannte Slab-Methode nach Kay [50, 64]. Hierbei werden die Ebenen nacheinander behandelt und der Abstand vom Strahlursprung zur ersten Ebene t^{min} und zur zweiten Ebene t^{max} berechnet (vgl. Abbildung 3.11).

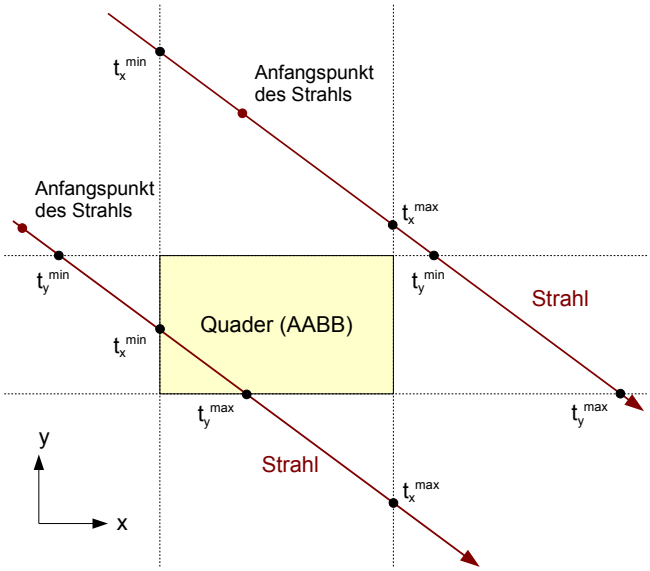


Abbildung 3.11: *Schnittpunkte Strahl/Quader*

Dies wird für die drei Achsen (x, y, z) durchgeführt und der größte Wert von t^{min} und der kleinste Wert von t^{max} gespeichert:

$$\begin{aligned} t^{min} &= \max(t_x^{min}, t_y^{min}, t_z^{min}) \\ t^{max} &= \min(t_x^{max}, t_y^{max}, t_z^{max}). \end{aligned} \tag{3.3}$$

Der Abstand vom Anfangspunkt des Strahls zum Schnittpunkt mit der ersten Ebene (entlang des Strahls) ist:

$$\begin{aligned} t_x^{min} &= (x_{b1} - x_1)/(x_2 - x_1) \\ t_y^{min} &= (y_{b1} - y_1)/(y_2 - y_1) \\ t_z^{min} &= (z_{b1} - z_1)/(z_2 - z_1) \end{aligned} \tag{3.4}$$

und für den Schnittpunkt mit der zweiten Ebene:

$$\begin{aligned} t_x^{max} &= (x_{b2} - x_1)/(x_2 - x_1) \\ t_y^{max} &= (y_{b2} - y_1)/(y_2 - y_1) \\ t_z^{max} &= (z_{b2} - z_1)/(z_2 - z_1). \end{aligned} \tag{3.5}$$

Für ein Quader mit den Eckpunkten x_{b1}, y_{b1}, z_{b1} und x_{b2}, y_{b2}, z_{b2}. Jetzt lässt sich sehr leicht überprüfen, ob der Strahl den Quader schneidet, in dem man t^{min} und t^{max} vergleicht. Ist t^{min} größer als t^{max} wird der Quader nicht geschnitten:

$$\begin{aligned} t^{min} > t^{max} &\Rightarrow \text{kein Schnittpunkt} \\ t^{min} \leq t^{max} &\Rightarrow \text{Schnittpunkt.} \end{aligned} \tag{3.6}$$

Schnittpunkte Strahl/Dreieck

Zahlreiche Verfahren zur Schnittpunktberechnung zwischen Strahl und Dreieck wurden in den letzten Jahrzehnten entwickelt [8, 80]. Ein allgemein gültiger Ansatz basiert auf der Darstellung des Dreiecks in baryzentrischen Koordinaten [3]. Ein Punkt $f(u, v)$ auf einem Dreieck ist hierbei gegeben durch:

$$f(u, v) = (1 - u - v)\, p_0 + u\, p_1 + v\, p_2 \tag{3.7}$$

wobei $u \geq 0,\ v \geq 0$ und $u + v \leq 1$ erfüllt sein müssen. Zur Berechnung der Schnittpunkte zwischen Strahl $r(t)$ und Dreieck $f(u, v)$ sind $r(t)$ und $f(u, v)$ proportional, daraus folgt:

$$\begin{gathered} o + td = (1 - u - v)\, p_0 + u\, p_1 + v\, p_2 \\ \Rightarrow \begin{pmatrix} -d & p_1 - p_0 & p_2 - p_0 \end{pmatrix} \begin{pmatrix} t \\ u \\ v \end{pmatrix} = o - p_0 \end{gathered} \tag{3.8}$$

Das Lösen dieser Gleichung liefert den gesuchten Schnittpunkt. Durch Translation des Dreiecks in den Ursprung und Transformation in ein Einheitsdreieck kann die Berechnung der Schnittpunkte beschleunigt werden, was zu Algorithmus 3.4 führt.

Algorithmus 3.4 Schnittpunkte Strahl/Dreieck

function CALCRAYTRIANGLEINTERSECTION(ray, tri)
  $e_1 = tri.p_1 - tri.p_0$
  $e_2 = tri.p_2 - tri.p_1$
  $q = ray.d \times e_2$
  $a = e_1 \cdot q$
  if a is equal to 0 **then**
    return no intersection
  end if
  $f = 1/a$
  $s = ray.o - tri.p_0$
  $u = f(s \cdot q)$
  if $u < 0$ **then**
    return no intersection
  end if
  $r = s \times e_1$
  $v = f(d \cdot r)$
  $w = 1 - u - v$
  if $u = 0$ *or* $v = 0$ *or* $w = 0$ **then**
    return intersect edge
  else if $v < 0$ *or* $u + v > 1$ **then**
    return no intersection
  else
    return intersection
  end if
end function

3.3.5 Benchmark des Kd-Baums

In diesem Abschnitt wird ein Benchmark der Kd-Baum Implementierung gezeigt. Hierbei wurden acht unterschiedliche Geometrien (vgl. Abbildung 3.12) mit variierenden Oberflächenauflösungen verwendet. Für jede Geometrie wurden zwei Kd-Bäume erstellt, einer mit dem Spatial Median Verfahren, bei dem die Unterteilung der Knoten in der geometrischen Mitte der Quader erfolgt und ein weiterer mit der heuristischen Kostenfunktion der Surface Area Heuristic (SAH). Für den Benchmark wurde ein einfacher Schnittpunkttest für 100.000 zufällig generierte Strahlen auf den erstellten Kd-Bäume durchgeführt. Die Ergebnisse sind in Tabelle 3.2 gegeben. Dargestellt sind die Anzahl der Oberflächendreiecke, die Knoten des Baums sowie der Speedup, der die Beschleunigung des SAH-Verfahrens gegenüber dem Spatial Median Verfahren angibt. Zu erkennen ist ein durchschnittlicher Speedup zwischen zwei und drei. Bei Testfällen wie einer komplexen Flugzeuggeometrie kann das SAH-Verfahren sein volles Potential ausnutzen, da es hier große Leerräume (Bereiche in denen keine Geo-

metrie vorhanden ist) gibt, die vom Baum maximiert werden und so zu einem Speedup von bis zu 7 führen.

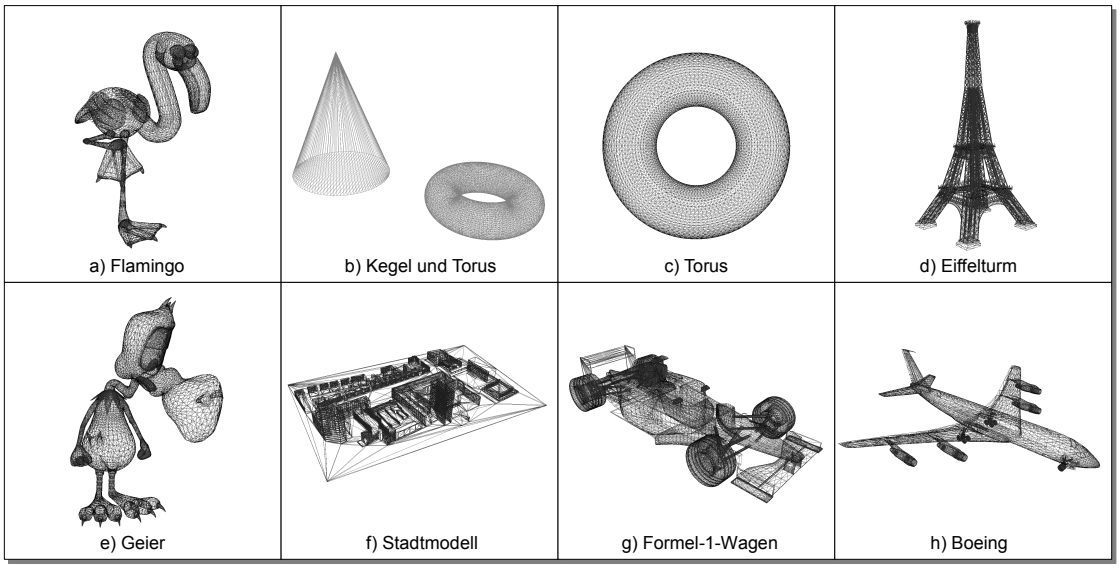


Abbildung 3.12: *Unterschiedliche Geometrien für Kd-Baum Benchmark*

Geometrie	**Dreiecke**	**Verfahren**	**Knoten**	**Speedup**
Flamingo a)	12861	SpatialMedian	8721	1,00
		SAH	50065	**3,17**
Kegel und Torus b)	12296	SpatialMedian	36514	1,00
		SAH	33240	**1,60**
Torus c)	12098	SpatialMedian	3889	1,00
		SAH	33829	**3,09**
Eiffelturm d)	12420	SpatialMedian	75318	1,00
		SAH	29321	**2,20**
Geier e)	24276	SpatialMedian	12691	1,00
		SAH	87053	**4,10**
Stadtmodell f)	73427	SpatialMedian	137088	1,00
		SAH	140925	**1,41**
Formel-1-Wagen g)	33478	SpatialMedian	126900	1,00
		SAH	140565	**3,29**
Boeing h)	46842	SpatialMedian	15211	1,00
		SAH	142490	**7,05**

Tabelle 3.2: *Kd-Baum Benchmark*

3.4 Gittergenerierung

Für die Diskretisierung der Bauteile werden anisotrope Knotengitter verwendet, die das Bauteil in diskrete Punkte (Knoten) zerlegen. Dieser Ansatz zur Beschreibung eines Volumenkörpers wird auch als Normzellen-Aufzählungsschema bezeichnet, bei dem die einzelnen Zellen des Gitters aus sogenannten Voxeln bestehen (vgl. Abbildung 3.13). Auf den Knotengittern wird die Differentialgleichung der Wärmeleitung in der Struktur mit der Finiten-Differenzen-Methode approximiert (vgl. Abschnitt 4.2). Die Struktur des Rechengitters ist von entscheidender Bedeutung für die Berechnungsdauer und Genauigkeit eines Lösungsverfahrens. Hierbei hängt die Effizienz des Verfahrens maßgeblich von den verwendeten Datenstrukturen der Knotenverteilungen ab. Ein performanter Ansatz bildet die Knoten direkt auf die Elemente einer Matrix ab, so dass benachbarte Knoten im physikalischen Raum auch benachbarte Elemente in der Matrix sind. Weitere Informationen über die Nachbarschaftstopologien (Verbindungen der Knoten zu den jeweiligen Nachbarknoten) wie bei unstrukturierten Gittern werden nicht benötigt. Somit lassen sich vor allem richtungsorientierte Verfahren (wie FDM) besonders effizient umsetzen. Im Folgenden wird ein effizienter und robuster Ansatz zur automatisierten Gittergenerierung vorgestellt, der durch seine kurze Rechenzeit ein interaktives Arbeiten ermöglicht. Mit diesem Ansatz lassen sich auch komplexe Gitter mit über 500^3 Knoten in wenigen Sekunden erstellen.

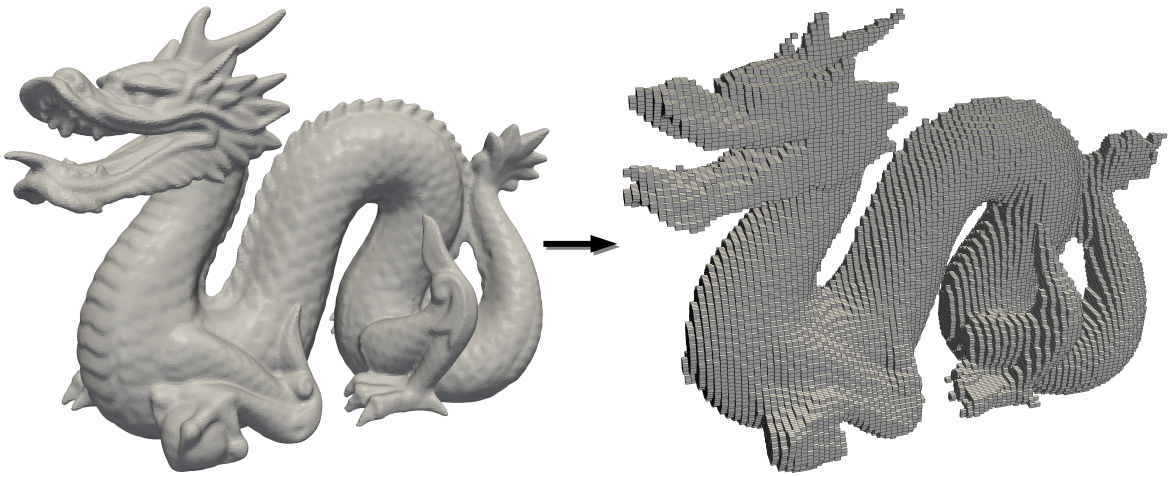


Abbildung 3.13: *Voxelisierung einer komplexen Geometrie (aus [43])*

3.4.1 Punkt-in-Polyeder-Test

Die Hauptroutine bei der Erstellung von Knotengittern ist der Punkt-in-Polyeder-Test zur Bestimmung, ob ein Knoten (Punkt) innerhalb oder außerhalb eines Polyeders liegt. Ein effizienter Ansatz hierzu ist der Ray-Crossing-Algorithmus (Strahlentest) nach O'Rourke

[89], der auf die meisten geometrischen Datensätze anwendbar ist [87]. Der Ray-Crossing-Algorithmus basiert auf dem Jordanschen Kurvensatz (Jordan Curve Theorem) [89], welcher besagt, dass jede geschlossene Kurve eine Ebene in zwei getrennte Gebiete zerlegt, deren gemeinsamer Rand die Kurve ist. Hieraus folgt, dass es für einen Strahl ausgehend von einem Punkt (in eine zufällige Richtung) eine bestimmte Anzahl an Schnittpunkten mit dem Rand gibt. Ist diese Anzahl der Schnittpunkte ungerade dann liegt der Punkt innerhalb des Polyeders, ist die Anzahl gerade dann liegt der Punkt außerhalb. Zusätzlich muss der Sonderfall beachtet werden, wenn der Strahl eine Kante oder Knoten trifft und die Anzahl der Schnittpunkte nicht eindeutig ist. In diesem Fall muss der Strahlentest für eine neue zufällige Richtung solange wiederholt werden bis er ein eindeutiges Ergebnis liefert. In 3.5 ist der allgemeine Ablauf dargestellt.

Algorithmus 3.5 Punkt-in-Polyeder-Test

```
function ISPOINTINPOLYHEDRON(p)
    while true do
        ray(p,d)                    ▷ Strahl ausgehend von Punkt p mit zufälliger Richtung d
        n = 0
        for each triangle do
            status = CALCRAYTRIANGLEINTERSECTION(ray, triangle)
            if status = intersect edge then
                Go back to while
            else if status = intersection then
                n = n + 1
            end if
        end for
        if n is odd then
            p is inside
        else
            p is outside
        end if
    end while
end function
```

Dieser Ansatz führt zu einer algorithmischen Komplexität von $O(n)$ (mit der Anzahl der Dreiecke n), da der Strahl gegen alle Dreiecke des Körpers getestet werden muss. Eine wesentliche Beschleunigung des Verfahrens kann unter Verwendung eines Raumpartitionierungsverfahren erfolgen. Hierbei haben sich, wie in Abschnitt 3.3 dargestellt, Kd-Bäume als optimale Datenstruktur zur Speicherung der Oberflächendreiecke erwiesen. Die Komplexität des Ray-Crossing-Algorithmus auf Kd-Bäumen reduziert sich auf $O(log\ n)$ bei linearem Speicherverbrauch.

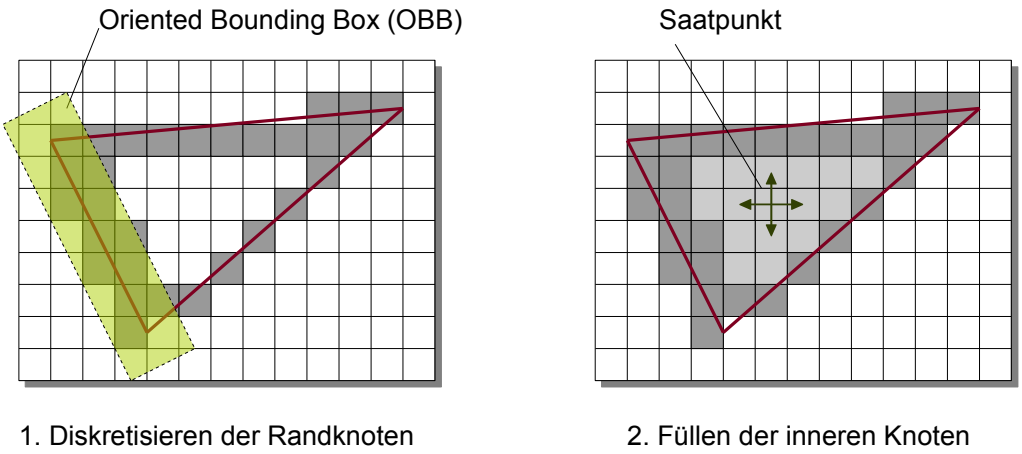


Abbildung 3.14: *Gittergenerierung in zwei Schritten*

3.4.2 Füllalgorithmus

Trotz der Optimierung des Verfahrens ist es immer noch mit hoher Laufzeit verbunden, den Ray-Crossing-Algorithmus für alle Knoten des Rechengitters durchzuführen. Aus diesem Grund wurde zur weiteren Beschleunigung des Verfahrens ein Füllalgorithmus verwendet. Die Gittergenerierung wird dann in zwei Teilschritten durchgeführt (vgl. Abbildung 3.14). Im ersten Schritt wird der Rand der Geometrie auf dem Knotengitter diskretisiert. Hierzu werden Hüllkörper (Oriented-Bounding-Box OBB) um die Primitiven der Geometrie (z.B. Linien in 2D oder Dreiecke in 3D) gelegt und nur die Knoten innerhalb der Hüllkörper mit dem Ray-Crossing-Algorithmus getestet. Somit braucht nur ein kleiner Teil des Knotengitters mit dem vergleichsweise aufwändigen Strahlentest berechnet werden.

Im zweiten Schritt kann, da der Rand der Geometrie nun eindeutig auf dem Gitter festgelegt ist, der innere Bereich mit einem Flutfüll-Algorithmus gefüllt werden. Bei der Flutfüllung (Floodfill) werden Flächen zusammenhängender Knoten mit einem neuen Wert belegt. Ausgangspunkt ist hierbei ein sogenannter Saatpunkt, der innerhalb der Fläche liegen muss, von dem aus die Nachbarknoten getestet werden, ob sie noch den alten Wert enthalten und werden ggf. mit dem neuen Wert belegt [25, 40, 41, 102]. Dieser Vorgang wird rekursiv oder iterativ fortgesetzt, bis ein Randknoten erreicht wird. Im Rahmen dieser Arbeit wurde eine iterative Variante des Algorithmus verwendet, da bei dieser keine Gefahr eines Stack-Überlaufs, im Vergleich zur Rekursion, bei großen Geometrien besteht. Dieser nutzt die Stapelspeicher (Stack) Datenstruktur, um die Nachbarknoten beim Füllen temporär zwischenzuspeichern. Algorithmus 3.6 beschreibt den allgemeinen Ablauf der Flutfüllung.

Andere Ansätze wie die Methode zur Voxelisierung nach Rueda und Ogayar [86, 98] ermöglichen eine schnelle Erstellung uniformer Gitter, ohne Verwendung besonderer Datenstrukturen. Da der Kd-Baum Voraussetzung für die effiziente Berechnung des Strahlungsaustau-

Algorithmus 3.6 Flutfüll-Algorithmus

```
function FLOODFILL(p)
    stack.push(x, y, z);
    while stack not empty do
        (x,y,z) = stack.pop()
        if (x,y,z) = FLUID then
            (x,y,z) = SOLID
            stack.push(x+1, y, z)
            stack.push(x-1, y, z)
            stack.push(x, y+1, z)
            stack.push(x, y-1, z)
            stack.push(x, y, z+1)
            stack.push(x, y, z-1)
        end if
    end while
end function
```

sches ist und somit als Datengrundlage vorliegt, wurden andere Verfahren nicht weiter betrachtet.

4 Simulation thermischer Transportvorgänge

Die Kontinuumsgleichungen zur physikalischen Beschreibung von thermischen Transportvorgängen (vgl. Kapitel 2) lassen sich häufig nur mit Verfahren der numerischen Mathematik lösen. Dieses Kapitel beschreibt einen effizienten numerischen Lösungsansatz zur verteilten Simulation des gekoppelten zeitabhängigen Problems von Wärmeleitung und Wärmestrahlung für 3d Umgebungen. Hierdurch lassen sich komplexe Problemstellungen aus unterschiedlichen Bereichen der Ingenieur- und Umweltwissenschaften mit kurzen Rechenzeiten berechnen. Im Rahmen dieser Arbeit wurde besonderer Fokus auf die effiziente algorithmische Umsetzung sowie Verfahren zur parallelen Berechnung auf Computerclustern und Grafikkarten (GPUs) gelegt.

Zur Lösung des Wärmestrahlungsproblems wird ein numerischer Ansatz, basierend auf der hierarchischen Radiosity-Methode verwendet, der den Strahlungsaustausch zwischen diffusen Oberflächen in einer abgeschlossenen Umgebung simuliert. Durch die Verwendung von optimierten Kd-Bäumen zur Speicherung der an der Strahlung beteiligten Objekte konnte die Laufzeitkomplexität deutlich reduziert werden [19, 20]. Der Strahlungskern ist direkt an ein Modul zur Simulation der Wärmeleitung gekoppelt. Hierbei wird der Energietransport in wärmeleitenden Materialien für instationäre Temperaturfelder mit einem Finite-Differenzen-Verfahren (FDM) berechnet. Dieses konnte aufgrund seiner besonderen Struktur effizient zur parallelen Berechnung auf Grafikkarten implementiert werden. Hierdurch lässt sich die typische Laufzeit um zwei Größenordnungen reduzieren.

4.1 Simulation thermischer Strahlung mit der Radiosity-Methode

Die Simulation thermischer Strahlungsprozesse erfordert die Berechnung des Strahlungsaustausches zwischen allen Oberflächen einer Umgebung, unabhängig davon wie weit diese voneinander entfernt sind. Die Menge der ausgetauschten Energie hängt hierbei von geometrischen Eigenschaften wie Größe, Distanz und Orientierung der Flächen sowie den Materialeigenschaften, der Verteilung der Energie auf die einzelnen Wellenlängen und dem Einfluss des Zwischenmediums ab. Diese mehrfache Abhängigkeit macht die Betrachtung von Strahlung schwierig. Eine Lösung des Strahlungsaustauschproblems für komplexe Umgebungen lässt sich häufig nur mit numerischen Näherungsverfahren finden. Hierzu sind in den letzten Jahrzehnten zahlreiche Methoden mit unterschiedlichen Anwendungsbereichen, Genauigkeiten und Laufzeiten entwickelt worden. Neben klassischen Verfahren wie der Finite-Elemente-Methode oder der Monte-Carlo-Simulation, die sich auf das Strahlungsproblem

anwenden lassen, sind spezielle Ansätze wie die Zonen-Methode, die Diskrete-Ordinate-Methode und die Momenten-Methode (P-N-Method) entstanden [79, 105].

Für Problemstellungen im Bauingenieurwesen, unter Annahme grauer diffus strahlender Oberflächen und der Vernachlässigung des Zwischenmediums, hat sich die auf dem Zonenmodell basierte Radiosity-Methode als besonders effizient erwiesen. Bei der Zonen-Methode (zonal method), vorgestellt 1958 von Hottel [55], lässt sich der Strahlungsaustausch zwischen isothermen Zonen (Flächen) in einer abgeschlossenen Umgebung durch Aufstellen der Energiebilanz über alle Zonen berechnen. Für jede Zone wird hierbei eine konstante Strahldichte für Reflexion und Absorption angenommen. Die Genauigkeit des Verfahrens hängt von der Anzahl der Zonen, mit denen die Oberflächen der Körper unterteilt werden, ab.

Eine Erweiterung der Radiosity-Methode unter Verwendung optimierter Kd-Baum-Datenstrukturen und adaptiver Verfeinerungstechniken wurde im Rahmen dieser Arbeit entwickelt und hat sich als sehr effiziente Methode zur Berechnung komplexer Anwendungen im Bauingenieurwesen erwiesen. In den folgenden Abschnitten wird die Simulation des Strahlungsaustausches mit der Radiosity-Methode vorgestellt. Hierbei werden neben den Grundlagen, ein hierarchischer Ansatz, eine Erweiterung für transluzente Körper und Verfahren zur parallelen Ausführung auf Computerclustern vorgestellt.

4.1.1 Die klassische Radiosity-Methode

Die Radiosity-Methode, basierend auf der Zonen-Methode [55], wurde in der Computergrafik aufgrund der Gleichartigkeit von Wärme- und Lichtstrahlen als Methodik zur globalen Beleuchtungsberechnung (rendering) entwickelt [48]. Mit dem Verfahren lässt sich der Strahlungsaustausch innerhalb einer abgeschlossenen Umgebung zwischen diffusen Oberflächen simulieren. Die Oberflächen werden hierbei in diskrete Elemente, die sogenannten Patches unterteilt. Patches sind in der Regel geometrische Primitive (wie Dreiecke oder Rechtecke). Basierend auf dem Energiegleichgewicht an einem Patch ergibt sich die ausgehende Strahlungsstromdichte (spezifische Ausstrahlung), die auch als Radiosity B bezeichnet wird, als Summe der ausgestrahlten und reflektierten Energie (vgl. Abbildung 4.1). Für eine konstante Strahlungsstromdichte über diskrete Patches folgt hieraus die Radiosity-Gleichung:

$$B_i = E_i + \rho \sum_{j=1}^{n} B_j F_{ij}, \tag{4.1}$$

mit der Eigenstrahlung E_i, dem Reflektionsgrad ρ (vgl. Abschnitt 2.2), dem Formfaktor F_{ij} und der Anzahl Patches der Umgebung n. Der Formfaktor beschreibt den Anteil der ausgetauschten Energie, in Abhängigkeit von Größe, Distanz und Orientierung der Flächen. Dieser hat einen Wert zwischen $0,0$ (keine Energie wird abgegeben) und $1,0$ (die Energie wird vollständig abgegeben) [121]. Die Herleitung der Formfaktors ist in Abschnitt 2.2.3 gegeben. Numerische Lösungsansätze werden in Abschnitt 4.1.2 erläutert.

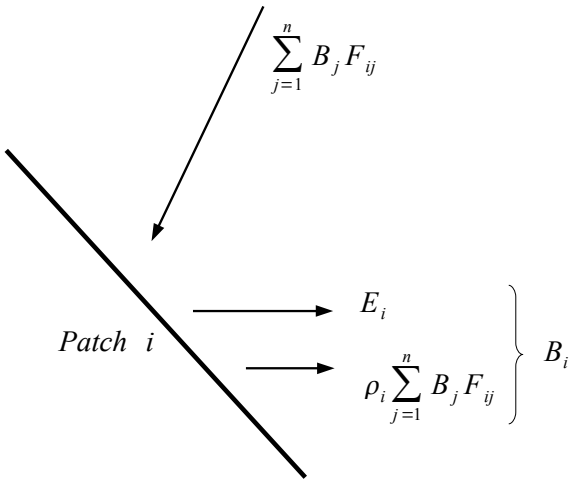


Abbildung 4.1: *Schematische Darstellung der Radiosity-Methode*

Für den vollständigen Strahlungsaustausch innerhalb einer abgeschlossenen Umgebung muss Gleichung 4.1 für alle n Patches berechnet werden. Diese n Gleichungen mit n Unbekannten ergeben das folgende lineare Gleichungssystem:

$$\begin{pmatrix} B_1 \\ B_2 \\ \dots \\ B_n \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ \dots \\ E_n \end{pmatrix} + \begin{pmatrix} \rho_1 F_{11} & \rho_1 F_{12} & \dots & \rho_1 F_{1n} \\ \rho_2 F_{21} & \rho_2 F_{22} & \dots & \rho_2 F_{2n} \\ \dots & \dots & \dots & \dots \\ \rho_n F_{n1} & \rho_n F_{n2} & \dots & \rho_n F_{nn} \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \dots \\ B_n \end{pmatrix}, \tag{4.2}$$

durch Umformung folgt

$$\begin{pmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \dots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \dots & -\rho_2 F_{2n} \\ \dots & \dots & \dots & \dots \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \dots & 1-\rho_n F_{nn} \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \dots \\ B_n \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ \dots \\ E_n \end{pmatrix}. \tag{4.3}$$

Das Gleichungssystem lässt sich in kompakter Schreibweise als Matrixoperation darstellen:

$$\hat{M}\vec{B} = \vec{E} \tag{4.4}$$

mit der sogenannten Formfaktormatrix M. Durch Lösen des Gleichungssystems ergibt sich die ausgehende Strahlungsstromdichte B_i für jeden Patch der Umgebung. Besonders effizient lässt sich das Gleichungssystem durch iterative Verfahren, mit geringem Rechenaufwand, lösen. Hierbei ist aufgrund der Diagonaldominanz der Formfaktormatrix M eine schnelle Konvergenz gesichert. Die Diagonaldominanz der Matrix ergibt sich aus zwei Eigenschaften: Erstens, dass eine ebene Fläche keine Energie an sich selbst abstrahlt (der Formfaktor also 0 ist). Somit können die Elemente auf der Hauptdiagonalen der Matrix zu 1 gesetzt werden und zweitens, dass die Summe jeder Matrixspalte (ohne Diagonalelement)

kleiner als 1 ist, da ein Patch nie mehr als seine maximale Ausstrahlung an die Umgebung abgeben kann.

Bei den iterativen Gleichungslösern werden in jedem Iterationsschritt die approximierten Werte des zuvor berechneten Schritts verwendet. Die Iteration verläuft bis zu einer vorgegebenen Fehlerschranke. Als Startwert für die unbekannte ausgehende Strahlungsstromdichte B_i kann hier als erste Näherung die Eigenstrahlung E_i der Flächen verwendet werden. Als effizientes iteratives Lösungsverfahren hat sich das Shooting-Verfahren, auch als progressive Verfeinerung (Progressive Refinement) bezeichnet, erwiesen [28]. Hierbei wird die unverteilte Energie eines Patches, basierend auf der Southwell-Iteration, auf die Flächen der Umgebung geschossen. Zu Beginn jedes Iterationsschritts werden die Patches nach der Größe der zu verteilenden Energie sortiert. Hierdurch zeichnen sich Verteilungspfade mit hoher Energie zuerst aus. Kleinere Energietransfers werden hinausgezögert bis diese Patches mehrmals bestrahlt wurden. Der Name progressive Verfeinerung ist entstanden, da mit Fortschreiten der Lösung das Ergebnis immer genauer wird. Ein Vorteil dieser Methode ist neben der schnellen Konvergenz, dass das System ohne Zwischenspeicherung der Formfaktoren gelöst werden kann und somit nur wenig Speicherplatz benötigt. Der Ablauf des Verfahrens ist in Algorithmus 4.1 dargestellt.

Algorithmus 4.1 Shooting-Verfahren

```
for each patch i do                          ▷ Initialisieren der Startwerte
    B_i = E_i
    ΔB_i = E_i                               ▷ ΔB_i ist die unverteilte Energie
end for
while not convergent do
    sort patches                             ▷ Patches absteigend nach ΔB sortieren
    for each patch i do
        for each patch j do
            rad = ΔB_i F_ij ρ_i
            ΔB_j = ΔB_j + rad
            B_j = B_j + rad
        end for
        ΔB_i = 0
    end for
end while
```

Die Radiosity-Methode kann für verschiedene Frequenzbereiche gelöst werden. Im Rahmen dieser Arbeit wurde eine Diskretisierung in einem kurzwelligen und einem langwelligen Bereich vorgenommen, so dass sich unter Verwendung von solaren Absorptionsgraden auch Sonneneinstrahlung und Reflexion an Bauteilen simulieren lässt.

Die klassische Radiosity-Methode hat eine algorithmische Komplexität von $O(n^3)$ und ist somit für die Berechnung des Strahlungsaustausches für komplexe Umgebungen schlecht ge-

eignet. Ein Problem stellt hierbei nicht das Lösen des Radiosity-Gleichungssystems dar, sondern die Berechnung der n^2 Formfaktoren. Effiziente Ansätze zur Berechnung der Formfaktoren sowie Beschleunigungstechniken zur Reduktion der Komplexität des Verfahrens werden in den folgenden Abschnitten vorgestellt.

4.1.2 Berechnung der Formfaktoren

Die Berechnung des Formfaktors F_{ij} (vgl. Abschnitt 2.2.3) hat großen Einfluss auf die Laufzeit und die Genauigkeit der Radiosity-Simulation. Ein großes Problem stellt hierbei die Lösung des im Formfaktor auftauchenden Doppelintegrals dar. Dies lässt sich häufig nur für einfache geometrische Anordnungen analytisch lösen, einige dieser Lösungen sind in Abschnitt 2.2.3 tabelliert. Ausführlichere Sammlungen von berechneten Formfaktoren kann u.a. Howell [57], Modest [79] oder Siegel et al. [105] entnommen werden. Für eine allgemeingültige Lösung komplexerer Geometrien sind numerische Ansätze nötig. Zahlreiche Arbeiten befassen sich mit Verfahren zur Lösung des Formfaktors [29, 91, 93, 105, 110]. Im Folgenden wird ein kurzer Überblick über mögliche Ansätze gegeben und die im Rahmen dieser Arbeit verwendete hierarchisch adaptive Methode erläutert.

Die in Abschnitt 2.2.3 eingeführte Formfaktorgleichung 2.16 wird für die Radiosity-Methode um eine binäre Sichtbarkeitsfunktion $V(p_i, p_j)$ zur Beschreibung der Sichtbarkeit zwischen den Flächen p_i und p_j erweitert. Hieraus folgt für den Formfaktor:

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\beta_i \, \cos\beta_j}{\pi r^2} \, V(p_i, p_j) \, dA_i \, dA_j, \tag{4.5}$$

mit

$$V(p_i, p_j) \quad \begin{cases} 1 \text{ falls } p_i \text{ und } p_j \text{ sichtbar zueinander sind} \\ 0 \text{ sonst} \end{cases}. \tag{4.6}$$

Der Kernel des Formfaktorintegrals wird als differenzielle Form des Formfaktors bezeichnet und entspricht der differenziellen Fläche zweier Patches:

$$F_{ij} = \frac{\cos(\beta_i)\cos(\beta_j)}{\pi r^2} \, V(p_i, p_j) \, dA_j. \tag{4.7}$$

Dies ist die einfachste Methode, den Formfaktor näherungsweise zu berechnen. Allerdings ist dieser Ansatz nur für weit entfernte Patches mit kleinen Oberflächen gültig, da er ansonsten zu Singularitäten führt.

Eine verbreitete Methode zur Lösung des Formfaktors ist die sogenannte Halbwürfel-Methode (Hemi-Cube), bei der ein Würfel um das Zentrum eines Patches gelegt wird, so dass die Flächennormale mit der Z-Achse des Hemi-Cubes übereinstimmt [29, 30]. Die Seitenflächen dieses Würfels werden in ein Raster aus diskreten Zellen unterteilt (vgl. Abbildung 4.2). Dabei hängt die Genauigkeit des Verfahrens direkt mit der gewählten Auflösung des Halbwürfels zusammen. Für jede Zelle wird ein so genannter Delta-Formfaktor, in Abhängigkeit der Zellposition und -orientierung, berechnet. Der Formfaktor lässt sich

nun durch Projektion aller Flächen auf die Zellen des Halbwürfel und anschließendes Aufsummieren aller Delta-Formfaktoren berechnen. Falls zwei Patches auf dieselbe Zelle projizieren, wird derjenige mit geringerem Abstand zum Mittelpunkt verwendet. Somit wird die Sichtbarkeitsberechnung zwischen Flächen realisiert. Ein Vorteil dieses Verfahrens ist die schnelle hardwarebeschleunigte Berechnung basierend auf dem Z-Buffer. Ein großer Nachteil der Halbwürfel-Methode sind, neben starken Aliasingeffekten bei zu geringen Auflösungen des Halbwürfels, die hohen Speicheranforderungen, die es für Simulationen komplexer Geometriemodelle unbrauchbar macht.

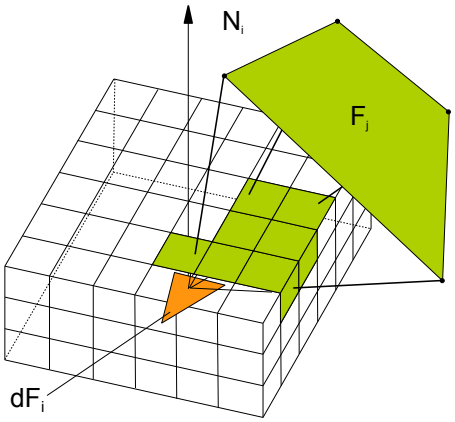


Abbildung 4.2: *Berechnung des Formfaktors durch Projektion eines Patches auf einen Halbwürfel (Hemi-Cube) [29]*

Ein Ansatz, der in dieser Arbeit verwendet wird, kombiniert eine adaptive Verfeinerung des Oberflächennetzes (vgl. Abschnitt 4.1.3) mit einer Kreisscheiben-Näherung (point-to-disk) des Formfaktors [120]. Durch diese Methode werden Singularitäten vermieden und es ist möglich, bei geringem Speicherbedarf komplexe Systeme zu simulieren. Bei der Kreisscheiben-Näherung wird die aussendende Fläche durch eine Kreisscheibe gleichen Flächeninhalts A_j approximiert. Somit ändert sich der Formfaktor zwischen einer differenziellen Fläche und einer beliebig orientierten Kreisscheibe zu (vgl. Abbildung 4.3):

$$F_{ij} = \frac{\cos(\beta_i)\cos(\beta_j)A_j}{\pi r^2 + A_i} V(p_i, p_j). \tag{4.8}$$

Dieses Verfahren wurde erstmals von Wallace 1989 [120] vorgestellt und hat sich als praktischer Näherungsansatz zur Lösung des Formfaktordoppelintegrals etabliert.

Eine weitere Möglichkeit den Formfaktor zu lösen, kann mittels stochastischer Verfahren (wie der Monte-Carlo-Methode) erfolgen. Dieser Ansatz wurde hier aufgrund relativ hoher Laufzeiten nicht weiter betrachtet und kann u.a. [91, 93] entnommen werden.

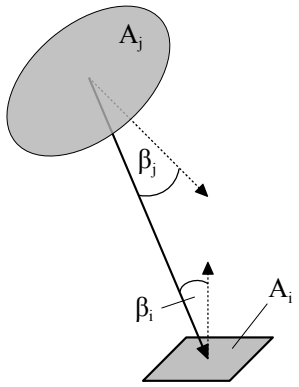


Abbildung 4.3: *Formfaktorberechnung durch Kreisscheiben-Näherung*

Lösen des Sichtbarkeitsproblems

Bei der Berechnung des Formfaktors muss neben der Lösung des doppelten Flächenintegrals auch die Sichtbarkeit $V(p_i, p_j)$ zwischen den Flächen berechnet werden. Ist die Sicht zwischen zwei Flächen durch andere Objekte verdeckt, können diese (im Fall von opaken Objekten) auch keine Energie untereinander austauschen. Die Berechnung des Sichtbarkeitsproblems ist somit wichtig und stellt den aufwändigsten Schritt bei der Formfaktorberechnung dar. Im Rahmen dieser Arbeit wurde ein effizienter Ansatz zur Bestimmung der Sichtbarkeiten zwischen Flächen auf Basis optimierter Kd-Bäumen zur Raumpartitionierung (vgl. Abschnitt 3.3) entwickelt. Für den Sichtbarkeitstest wird ein Strahl durch die Mittelpunkte zweier Patches gelegt. Für alle anderen Flächen der Umgebung wird dann geprüft, ob sie den Strahl schneiden (diese Strahlenverfolgung wird als Raytracing, Raycasting oder Rayshooting bezeichnet). Im Falle eines Schnittpunkts wird die ausgetauschte Energie abhängig von Transmissionsgrad τ der Fläche abgemindert, oder bei einer opaken Fläche (mit $\tau = 0$) zu null gesetzt. Hierzu wurde der im Formfaktor auftauchende binäre Sichtfaktor $V(p_i, p_j)$ erweitert, so dass unterschiedliche Transmissiongrade berücksichtigt werden können. Somit liefert die Sichtbarkeitsfunktion nicht mehr einen binären Wert (zwei Flächen sind sichtbar $V = 1$ oder nicht sichtbar $V = 0$ zueinander) sondern einen Wert zwischen $0,0 < V < 1,0$, der angibt wie viel Energie zwischen den beiden Flächen ausgetauscht wird. Die Erweiterung des rekursiven Sichtbarkeitstest (vgl. Algorithmus 3.3) für Transmissionsgrade ist in Algorithmus 4.2 dargestellt. Die Überprüfung der Sichtbarkeit resultiert in einer algorithmischen Komplexität von $O(n)$ und reduziert sich durch Ausnutzung der Kd-Baum Datenstruktur auf $O(log n)$, wobei n die Anzahl der Patches ist. Um die Genauigkeit des Verfahrens zu erhöhen und teilweise Abschattungen zwischen Flächen zu realisieren, können mehrere Teststrahlen verwendet werden. Hierbei werden zwischen verschiedenen Punkten auf den Patches (z.B. den Eckpunkten) Strahlen angenommen und ein Mittelwert aus den einzelnen Schnittpunktberechnungen gebildet. Häufig ist allerdings die Verwendung eines Strahls ausreichend, da die Flächen beim Strahlungsaustausch adaptiv verfeinert werden und somit Fehler durch teilweise Abschattungen gering sind.

Algorithmus 4.2 Hierarchischer Sichtbarkeitstest für Transmission

```
function VISIBILITY(patch_1, patch_2)
    create new mailbox
    return INTERSECTRAY(patch_1.center, patch_2.center, kd-tree root node, mailbox)
end function

function INTERSECTRAY(point_1, point_2, kdnode, mailbox)
    if bounding box (AABB) of kdnode intersects ray from p_1 to p_2 then
        if kdnode is no leaf node then
            passedEnergy1 = 1.0
            passedEnergy2 = 1.0
            if kdnode has left child c_l then
                passedEnergy1 = INTERSECTRAY(point_1, point_2, c_l, mailbox)
            end if
            if kdnode has right child c_r then
                passedEnergy2 = INTERSECTRAY(point_1, point_2, c_r, mailbox)
            end if
            return passedEnergy1 · passedEnergy2
        else
            passedEnergy = 1.0
            for each patch p inside the kdnode do
                if mailbox not contains p then
                    put p into mailbox
                    if p intersects ray from p_1 to p_2 then
                        passedEnergy = passedEnergy · τ_p
                    end if
                end if
            end for
        end if
    else
        return 1.0                          ▷ Energie wird vollständig durchgelassen
    end if
end function
```

4.1.3 Die hierarchische Radiosity-Methode

Der hierarchische Radiosity Ansatz, vorgestellt von Hanrahan 1991 [52], verwendet eine adaptive Unterteilung des Oberflächennetzes in Abhängigkeit der Formfaktoren bzw. der daraus resultierenden Energiegradienten. Dieser Ansatz, abgeleitet von der Lösung des N-Körperproblems (N-Body problem) [10], reduziert die Komplexität der klassischen Radiosity-Methode von $O(n^2)$ (ohne Sichtbarkeitsprüfung, alle Patches stehen in Interaktion miteinander) auf $O(k^2 + n)$, mit der Anzahl der groben Eingangsflächen k und der Anzahl der verfeinerten Flächen n. Grundidee der hierarchischen Radiosity-Methode ist die Verfeinerung von Patches, zwischen denen der Formfaktor einen Grenzwert übersteigt und zu einem unannehmbaren Fehler der Gesamtlösung führen würde. Hierbei wird solange eine Unterteilung vorgenommen, bis der Formfaktor unter einen Schwellwert fällt.

Der Algorithmus beginnt mit der Berechnung der $k(k-1)/2$ Formfaktoren zwischen allen groben Eingangspatches. Wird hierbei der Grenzwert für den Formfaktor überschritten, werden die beiden beteiligten Patches in jeweils vier feinere Patches unterteilt. Dieser Vorgang wird rekursiv fortgesetzt. Für jeden ursprünglichen Patch werden hierbei die Verbindungen zu seinen unterteilten feineren Patches in einer Quadtree ähnlichen Hierarchie abgelegt. Durch diesen Ansatz wird die Formfaktormatrix der klassischen Radiosity-Methode durch eine hierarchische Darstellung ersetzt. Die hierarchische Radiosity-Methode ist in Algorithmus 4.3 gegeben.

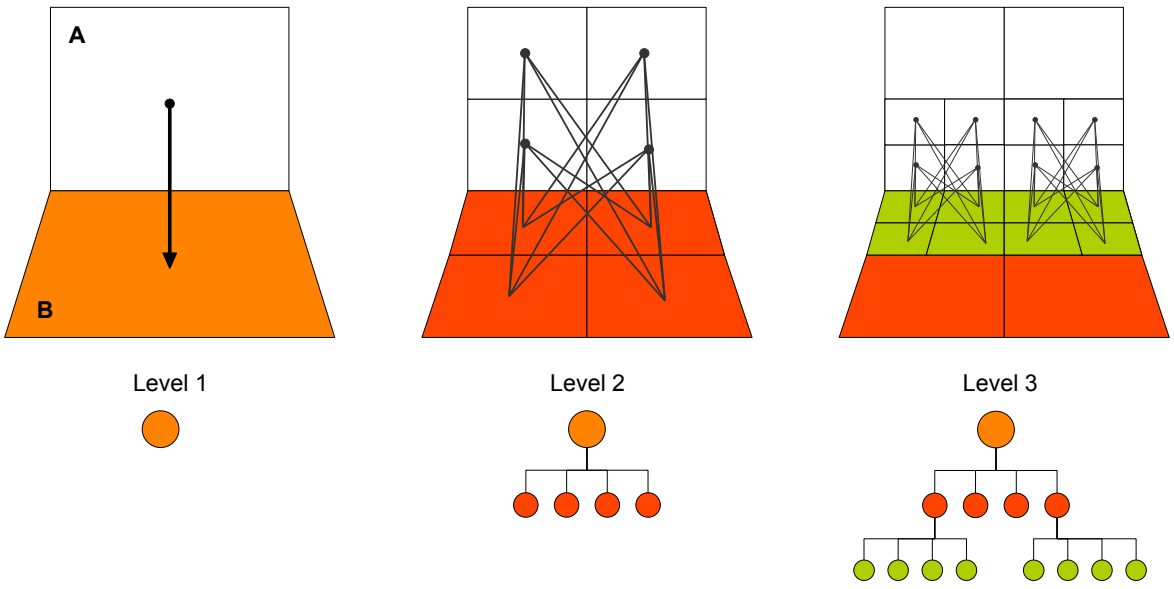


Abbildung 4.4: *Quadtree ähnliche hierarchische Unterteilung der Fläche B für Formfaktoren die den Grenzwert überschreiten*

In Abbildung 4.4 ist der Energieaustausch zwischen zwei rechteckigen Patches A und B dargestellt. Der Formfaktor auf oberster Ebene zwischen den beiden ursprünglichen Flächen ist

groß, so dass die beiden Flächen in jeweils vier kleinere unterteilt werden. Von den im nächsten Schritt berechneten 16 Formfaktoren überschreiten nur die vier Flächen an der gemeinsamen Kante den Grenzwert, die dann weiter unterteilt werden. Je nach vorgegebener Genauigkeit wird dieser Prozess rekursiv fortgeführt. Unterschreitet der Formfaktor zwischen zwei Flächen den Grenzwert (ist die Verfeinerung also ausreichend für die vorgegebene Genauigkeit) wird für den Energieaustausch zwischen diesen beiden Flächen eine Verbindung, ein sogenannter Link, eingerichtet. Solche Links können sich auf unterschiedlichen Ebenen der Hierarchie ergeben. Ein Energieaustausch findet also nicht nur zwischen den Blättern statt, sondern auch zwischen internen Knoten. Nachdem alle Verbindungen aufgebaut wurden und die Unterteilung der Patches abgeschlossen ist, kann für jeden Knoten die Energie über seine eingehenden Verbindungen (Links) eingesammelt und mit dem zugehörigen Formfaktor multipliziert werden. Dieser Prozess wird auch als gathering bezeichnet.

Die Verbindungen können auf allen Ebenen der Hierarchie aufgebaut sein. Aus diesem Grund muss die Energie, nach dem Einsammeln, an jedem Knoten (im Quadtree) nach unten zu den Blättern gedrückt und anschließend die gewichtete Energie wieder nach oben gezogen werden. Dieser Vorgang, auch als push-pull bezeichnet, ist in Algorithmus 4.5 gegeben.

Algorithmus 4.3 Hierarchische Radiosity-Methode

```
// Initiale Unterteilung der Flächen
for each patch patch_1 do
    for each patch patch_2 do
        REFINE(patch_1, patch_2)                    ▷ vgl. Algorithmus 4.4
    end for
end for

// Lösen des Systems
while not convergent do
    for each patch patch do
        GATHER(patch)                               ▷ vgl. Algorithmus 4.5
    end for
    for each patch patch do
        PUSHPULL(patch, 0.0)                        ▷ vgl. Algorithmus 4.5
    end for
end while
```

Im Folgenden ist noch einmal kurz der allgemeine Ablauf der hierarchischen Radiosity-Methode bestehend aus drei Schritten dargestellt:

1. Initiale Unterteilung der Flächen (initial linking): Der Formfaktoren wird zwischen allen groben Eingangspatches berechnet, hierzu wird die Kreisscheiben-Näherung (vgl. Gleichung 4.8) verwendet. Für jeden Formfaktor, der größer als ein vorgegebe-

ner Formfaktorgrenzwert (F_ε) und ein Flächengrenzwert A_ε ist (abhängig von der benötigten Genauigkeit der Lösung), wird eine Unterteilung des Patches mit der größeren Fläche vorgenommen. Andernfalls wird ein sogenannter Link zwischen den beiden Flächen angelegt, der für den Energieaustausch verwendet wird. Jeder Unterteilungsschritt erhöht hierbei die Genauigkeit der Lösung.

2. Einsammeln der Energie (energy gathering): Für jeden Patch wird die Energie über seine eingehenden Links eingesammelt, bis ein Konvergenzkriterium erreicht ist.

3. Push-pull: Eingehende Energie an einem Patch muss durch die gesamte Patch Hierarchie propagiert werden um einen konsistenten Zustand nach jedem Iterationsschritt zu erhalten. Hierzu wird die Energie des oberen Knoten (im Quadtree) nach unten zu den Blättern gedrückt und anschließend die gewichtete Energie wieder nach oben gezogen.

Algorithmus 4.4 Hierarchische Verfeinerung

procedure REFINE($patch_1$, $patch_2$)
  F_{12} = VISIBILITY($patch_1$, $patch_2$) $\cdot \frac{\cos(\Theta_1)\cos(\Theta_2) area_2}{\pi r^2 + area_1}$ ▷ Formfaktorberechnung

  if $F_{12} < F_{crit}$ or $area_1 < area_\epsilon$ or $area_2 < area_\epsilon$ **then**
    LINK($patch_1$, $patch_2$)
  else
    if $area_1 \geq area_2$ **then**
      SUBDIVIDE($patch_1$)
      for all children c_i of $patch_1$ **do**
        REFINE(c_i,$patch_1$)
      end for
    else if
      thenSUBDIVIDE($patch_2$)
      for all children c_i of $patch_2$ **do**
        REFINE(c_i,$patch_2$)
      end for
    end if
  end if
end procedure

Bei der Verfeinerung des Oberflächennetzes müssen die neuen Dreiecke optimal angeordnet (vernetzt) werden um bestmögliche Ergebnisse beim Strahlungsaustausch zu erhalten. Im Rahmen dieser Arbeit werden ausschließlich Dreiecksnetze betrachtet. Die Arbeiten von Durand [37], Schäfer [100] und Shaw [103] gehen im Detail auf effiziente Strategien zur Vernetzung (meshing) beim hierarchischen Radiosity ein. Im Folgenden wird ein solcher effizienter Ansatz kurz vorgestellt. Beim hierarchischen Radiosity werden Dreiecke in vier feinere

Algorithmus 4.5 Einsammeln (gathering) der Energie und push-pull

```
procedure GATHER(patch)
    B = 0.0
    for each incoming link l do
        B = B + l.F_12 · l.ΔB · α                    ▷ absorbierte Energie
    end for
end procedure
procedure PUSHPULL(patch,B_down)
    B_up = 0.0
    if patch has children then
        for all children c_i of patch do
            B_up = B_up + area_i/area· PUSHPULL(c_i, patch.B + B_down)
        end for
    else if
        thenB_up = B + B_down
    end if
    B = B_up
    return B_up
end procedure
```

Dreiecke unterteilt; ein verbreitetes Schema zur Unterteilung ist in Abbildung 4.5a dargestellt. Dieses Verfahren kann zu sogenannten hängenden Knoten (T-vertices) führen. Dies sind Eckpunkte verfeinerter Dreiecke im inneren eines Netzes, die keinen Eckpunkt für ein anderes Dreieck bilden. T-vertices sind ein häufig auftretendes Problem bei der Verfeinerung oder Vergröberung von Oberflächennetzen. Um numerische Probleme und diskontinuierliche Netzübergänge zu vermeiden, muss eine Elimination dieser Knoten durchgeführt werden. Hierzu werden in jedem rekursiven Schritt der Unterteilung alle Knoten getestet. Jeder hängende Knoten wird dann mit dem nächsten Eckpunkt des Nachbardreiecks verbunden. Abbildung 4.5b, 4.5c, und 4.5d stellen die stufenweise Anpassung des Netzes dar. Eine effiziente Methode zur Elimination von T-vertices kann der Arbeit von Schäfer [100] entnommen werden.

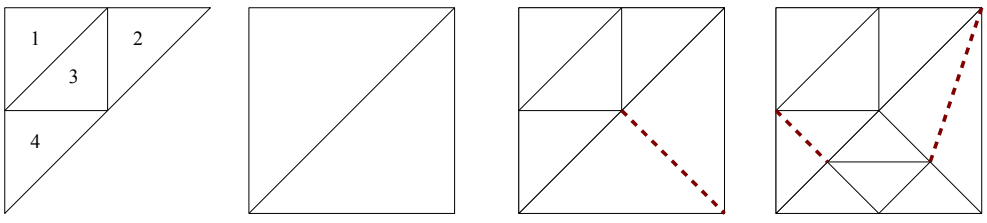


Abbildung 4.5: *a) Unterteilung der Dreiecke, b-d) T-vertex Elimination*

4.2 Simulation der Wärmeleitung mit Finite-Differenzen

Das zeitabhängige Wärmeleitungsproblem (vgl. Abschnitt 2.3.2) für komplexe Geometrien und Randbedingungen lässt sich häufig nicht analytisch lösen. Numerische Methoden stellen in vielen Fällen die einzige Möglichkeit zur Lösung der Wärmeleitungsgleichung dar. Im Rahmen dieser Arbeit wurde hierzu ein explizites Finite-Differenzen-Verfahren zur Simulation des Energietransport in wärmeleitenden Materialien entwickelt und an den Strahlungskern gekoppelt. Dieses Verfahren konnte aufgrund seiner besonderen Struktur effizient zur parallelen Berechnung auf Grafikkarten implementiert werden, wodurch die Laufzeit deutlich reduziert wird. Im Folgenden wird die näherungsweise Lösung der Wärmeleitungsgleichung mit Finite-Differenzen sowie den benötigten Randbedingungen erläutert. Abschließend wird der hardware-beschleunigte Ansatz auf Basis der Compute Unified Device Architecture (CUDA) von Nvidia dargestellt [84].

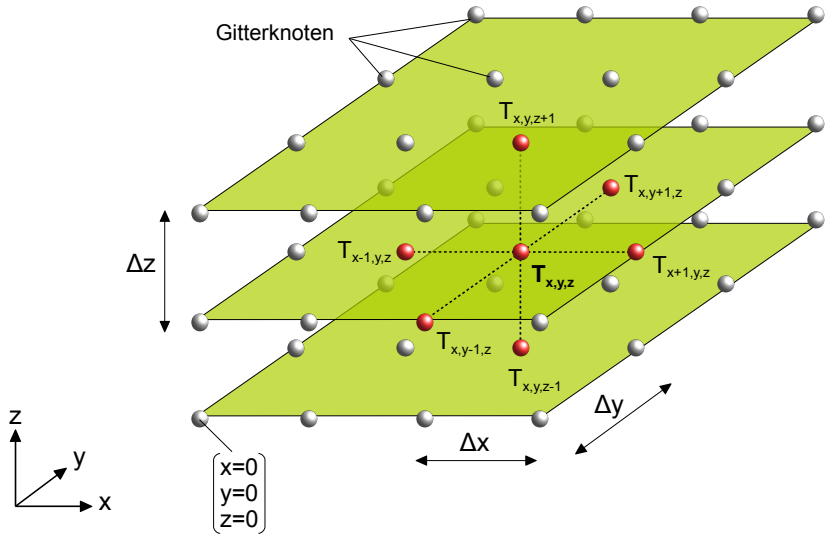


Abbildung 4.6: *3D Gitter zur räumlichen Diskretisierung der Wärmeleitungsgleichung*

4.2.1 Der FDM-Ansatz

Bei der Finite-Differenzen-Methode werden die Ableitungen $\partial T/\partial t$, $\partial^2 T/\partial x^2$, $\partial^2 T/\partial y^2$ und $\partial^2 T/\partial z^2$ in der Wärmeleitungsgleichung (vgl. Gleichung 2.33) durch Differenzenquotienten ersetzt. Hierdurch wird die Differentialgleichung näherungsweise auf einem diskreten Gitter in Raum und Zeit gelöst. Die Zeitableitung wird hierbei durch einen vorderen Differenzenquotienten für Zeitschritte $t \geq 0$ mit einer Schrittweite Δt ersetzt (vgl. Abbildung 4.7). Ein weiterer Ansatz unter Verwendung eines expliziten Runge-Kutta-Verfahren, zur näherungsweisen Berechnung der Zeitableitung mit höherer Ordnung, wird im Rahmen dieser Arbeit nicht weiter betrachtet [51, 96].

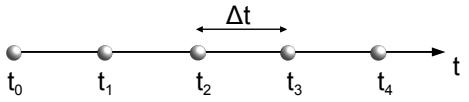


Abbildung 4.7: *Zetiliche Diskretisierung der Wärmeleitungsgleichung*

Der Raum wird durch ein anisotropes kartesisches 3D Gitter im Bereich $x_0 \leq x \leq x_{nx}$, $y_0 \leq y \leq y_{ny}$ und $z_0 \leq z \leq z_{nz}$ diskretisiert, mit der Anzahl Knoten nx, ny und nz in die jeweilige Achsrichtung (ein effizienter Ansatz zur Gittergenerierung ist in Abschnitt 3.4 erläutert) (vgl. Abbildung 4.6). Somit können die zweiten Ableitungen im Raum auf diesem Gitter durch zentrale Differenzen ersetzt werden. Hierbei kann durch Wahl kleinerer Abstände der Gitterpunkte (Δx, Δy und Δz) eine höhere Genauigkeit der Lösung erreicht werden. Dies geschieht allerdings zu Lasten des Rechenaufwands.

Aus Gleichung 2.33 (ohne innere Quellterme) folgt dann die Differenzengleichung:

$$\begin{aligned} \frac{T^{t+1}_{x,y,z} - T^{t}_{x,y,z}}{\Delta t} &= a\left(\frac{T^{t}_{x+1,y,z} - 2T^{t}_{x,y,z} + T^{t}_{x-1,y,z}}{\Delta x^2}\right. \\ &+ \left.\frac{T^{t}_{x,y+1,z} - 2T^{t}_{x,y,z} + T^{t}_{x,y-1,z}}{\Delta y^2} + \frac{T^{t}_{x,y,z+1} - 2T^{t}_{x,y,z} + T^{t}_{x,y,z-1}}{\Delta z^2}\right) \end{aligned} \tag{4.9}$$

und durch Umformen die Approximation für den neuen Zeitschritt $t+1$:

$$\begin{aligned} T^{t+1}_{x,y,z} &= T^{t}_{x,y,z} - 2a\Delta t\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\right)T^{t}_{x,y,z} \\ &+ a\Delta t\left(\frac{T^{t}_{x+1,y,z} + T^{t}_{x-1,y,z}}{\Delta x^2} + \frac{T^{t}_{x,y+1,z} + T^{t}_{x,y-1,z}}{\Delta y^2} + \frac{T^{t}_{x,y,z+1} + T^{t}_{x,y,z-1}}{\Delta z^2}\right) \end{aligned} \tag{4.10}$$

mit der Temperaturleitfähigkeit a (vgl. Gleichung 2.32). Somit lässt sich die Temperatur an einem Knoten im Inneren des Körpers in Abhängigkeit seiner sechs direkten Nachbarknoten berechnen. Das Verfahren ist räumlich zweiter Ordnung genau. Der Diskretisierungsfehler geht somit bei Verkleinerung der Gitterabstände quadratisch gegen null. Die zeitliche Konvergenzordnung beträgt Δt.

Das verwendete Differenzenverfahren ist numerisch nur bedingt stabil. Das bedeutet, dass der Fehler für Zeitschrittweiten die größer als ein Grenzwert sind ($\Delta t > \Delta t_\varepsilon$) nicht im Verlauf der Rechnung abklingt. Gleichung 4.10 bleibt für gegebene Gitterabstände Δx, Δy und Δz nur stabil, wenn die Bedingung:

$$1 - 2\frac{a\Delta t}{\Delta x^2} - 2\frac{a\Delta t}{\Delta y^2} - 2\frac{a\Delta t}{\Delta z^2} \geq 0, \tag{4.11}$$

erfüllt ist [9]. Hieraus folgt durch Umformen die Stabilitätsbedingung mit:

$$\Delta t \leq \frac{1}{2a}\ \frac{1}{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}}. \tag{4.12}$$

Im Rahmen dieser Arbeit wird auf den Rechengittern immer die größtmögliche Zeitschrittweite mit:

$$\Delta t = \frac{1}{2a} \; \frac{1}{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}} \; c \in [0.9, 1.0[\tag{4.13}$$

gewählt, so dass Δt unbedingt stabil bleibt und der Rechenaufwand verringert wird.

4.2.2 Diskretisierung der Randbedingungen

Gleichung 4.10 gilt nur für innere Knoten des Gitters, mit mindestens einem Nachbarknoten pro Achsrichtung. An den Ränder mit x_0, x_{nx}, y_0, y_{ny} und z_0, z_{nz} müssen gesonderte Randbedingungen betrachtet werden. Hierbei wird zwischen drei unterschiedlichen Randbedingungen unterschieden (vgl. Abschnitt 2.3.3). Im Rahmen dieser Arbeit werden die Neumann-Randbedingung, zur Vorgabe von Wärmeflüssen an der Körperoberfläche, sowie die Wärmeübergangsbedingung zum Fluid (Randbedingung der dritten Art) verwendet. Für die Vorgabe von Wärmeflüssen (vgl. Gleichung 2.36) wird die Ableitung in Richtung der Flächennormalen $\vec{n}$ durch eine Vorwärtsdifferenz ersetzt. Hieraus folgt exemplarisch für den linken Rand an der Stelle $x = 0$:

$$T^{t+1}_{x=0,y,z} = T^{t}_{x=0,y,z} + a\Delta t \frac{1}{\Delta x^2} \left(-\frac{7}{2} T^{t}_{x=0,y,z} + \frac{8}{2} T^{t}_{x=1,y,z} - \frac{1}{2} T^{t}_{x=2,y,z} \right) - \frac{\Delta t \; a \, 3 \dot{q}_W \; \vec{n}_x}{\lambda \Delta x}, \tag{4.14}$$

mit der Wärmestromdichte $\dot{q}_W$ am Rand. Für $\dot{q}_W = 0$, gilt die Gleichung 4.14 für adiabate, also perfekt isolierte, Ränder. Analog hierzu lassen sich die Randbedingungen für die übrigen Ränder aufstellen. Die Strahlungs-Struktur-Wechselwirkung, also die Kopplung von Wärmestrahlung und Wärmeleitung, kann somit über die Vorgabe von Wärmeflüssen an der Körperoberfläche realisiert werden. Unter Berücksichtigung der Zeitschrittweiten Δt_W und Δt_S beider Verfahren kann nun an der Oberfläche der Körper die Ein- und Ausstrahlung berechnet werden, die dann als Wärmestromdichte am Rand bei der Wärmeleitung berücksichtigt wird. Hierzu sind den Randknoten des Rechengitters die zugehörigen Oberflächendreiecke zugeordnet (vgl. Abbildung 4.8). Bei deutlich geringeren Auflösungen des Finite-Differenzen Gitters gegenüber der Dreiecksvernetzung kann dies gegebenenfalls zu einer Reduktion der Konvergenzordnung des gekoppelten Problems führen.

Für den Wärmeübergang zwischen Festkörper und Fluid wurde eine vereinfachte Übergangsbedingung mittels des Wärmeübergangskoeffizienten α_{WF} realisiert (vgl. auch Abschnitt 2.3.3). Hierzu wurde ein zusätzlicher Knoten außerhalb des Bauteils im Fluid verwendet, an dem eine Umgebungstemperatur T_F gesetzt wird (vgl. Abbildung 4.9). Der Wärmeübergangskoeffizient ist eine spezifische Kennzahl einer Konfiguration und kann für unterschiedliche Windgeschwindigkeiten der DIN EN ISO 6946 [35] entnommen werden (siehe hierzu auch Fischer et al. [39]).

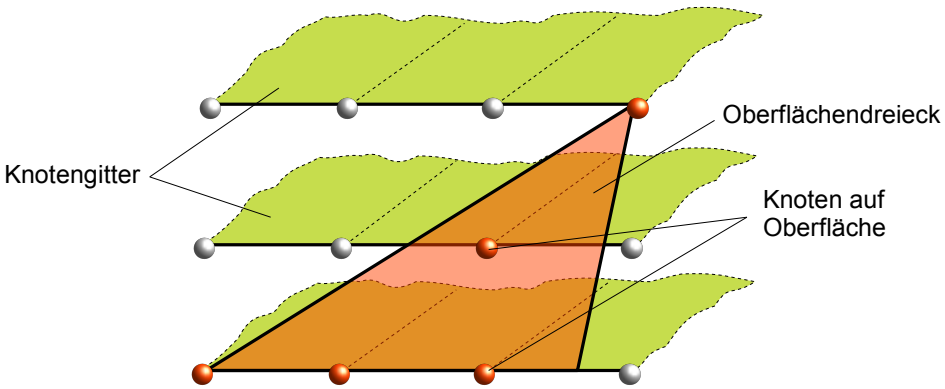


Abbildung 4.8: *Kopplung zwischen Oberflächendreieck und Knotengitter*

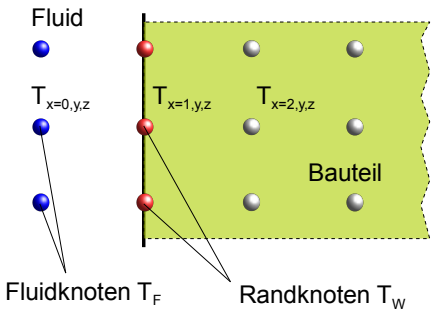


Abbildung 4.9: *Wärmeübergangsbedingung zwischen Fluid und Festkörper*

Die räumliche Ableitung in Richtung der Flächennormalen $\vec{n}$ wird bei der Wärmeübergangsbedingung (vgl. Gleichung 2.40) durch eine zentrale Differenz ersetzt. Hieraus folgt exemplarisch für den linken Rand die Differenzengleichung:

$$T^{t+1}_{x=1,y,z} = T^{t}_{x=1,y,z} - 2\frac{a\Delta t}{\Delta x^2}\left(1+\frac{\alpha_{WF}\Delta x}{\lambda}\right)T^{t}_{x=1,y,z} + 2\frac{a\Delta t}{\Delta x^2}T^{t}_{x=2,y,z} + 2\frac{a\Delta t\alpha_{WF}}{\Delta x\lambda}T^{t}_{x=0,y,z} \tag{4.15}$$

mit

$$\begin{aligned} T^{t}_{x=0,y,z} &= T_F \quad \text{(Temperatur des Fluids)} \\ T^{t}_{x=1,y,z} &= T_W \quad \text{(Temperatur der Wandoberfläche).} \end{aligned} \tag{4.16}$$

4.2.3 Kopplung von Bauteilen

Der Wärmefluss zwischen aneinandergrenzenden Bauteilen findet über ihre Kontaktfläche statt. Für die Kopplungsbedingung dieser Bauteile gilt, dass der Temperaturverlauf an den Kontaktflächen keine Unstetigkeiten aufweisen darf. Aufgrund der unterschiedlichen Diskretisierung der Bauteile sind für die Kopplung räumliche Interpolationen notwendig. Da die Randknoten durch die unterschiedlichen Gitterabstände auf den einzelnen Rechengittern nicht immer direkt aufeinanderliegen, werden die Temperaturwerte der Nachbarknoten

linear interpoliert. Der Wärmeübergang zwischen den Bauteilen mit den benötigten Interpolationen wird über sogenannte Interfaces realisiert. Ein Interface verbindet hierbei immer zwei Bauteile entlang einer gemeinsamen Kontaktfläche. Durch diese Trennung benötigt das Bauteil keine weiteren Informationen über seine benachbarten Bauteile. Die notwendigen Randbedingungen und Interpolationsalgorithmen sind vollständig in das Interface ausgelagert. Das Interface besteht aus einer Knotenreihe (Ebene) mit der Auflösung dy und dz des feineren Gitters (vgl. Abbildung 4.10). Auf den Gittern der Bauteile werden die Randknoten am Interface als Interfaceknoten markiert. Für diese Randknoten wird auf den Bauteilen kein Temperaturwert berechnet. Die Temperaturen werden stattdessen auf dem Interface berechnet und nach jedem Zeitschritt zurück an die Bauteilgitter kopiert.

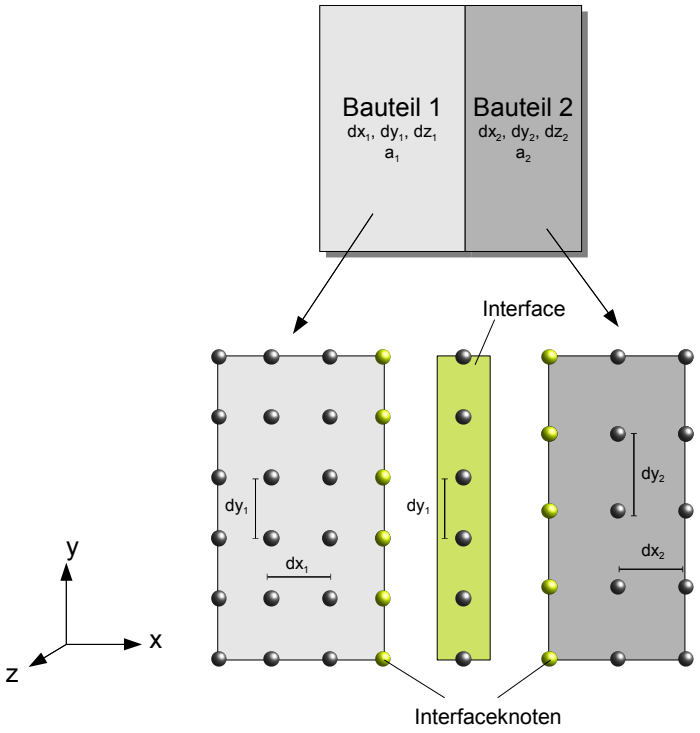


Abbildung 4.10: *Kopplung aneinandergrenzender Bauteile über Interfaces*

Der generelle Ablauf in jedem Zeitschritt der Berechnung erfolgt in drei Teilschritten und ist für zwei Bauteile mit gemeinsamer Kontaktfläche entlang der y-z-Ebene und geringerer Gitterauflösung des linken Bauteils in Abbildung 4.13 dargestellt. Zu Beginn erfolgt die Berechnung der Temperatur für den aktuellen Zeitschritt parallel auf beiden Bauteilen, unter Verwendung der in den vorherigen Abschnitten vorgestellten Finite-Differenzen-Ausdrücke und Randbedingungen (vgl. Abschnitt 4.2.1 und Abschnitt 4.2.2). Hierbei werden alle Gebiets- und Randknoten, mit Ausnahme der Gitterknoten an der Kontaktfläche (den Interfaceknoten), aktualisiert.

Im zweiten Schritt wird die Berechnung der Temperaturwerte auf dem Interface durchgeführt. Hierzu wird ein aus Gleichung 4.9 abgeleiteter leicht veränderter FDM-Stern auf dem

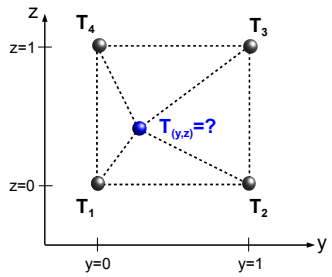
Interface verwendet, so dass sich unterschiedliche Temperaturleitfähigkeiten a_1 und a_2 auf den Bauteilen realisieren lassen. Bei diesem Ansatz basierend auf der Erhaltung der Flüsse am Interface, werden die Flüsse zwischen den Bauteilen 1 und 2 betrachtet:

$$\begin{aligned} \dot{q}_{2\to 1} &= -a_2 \nabla T \\ \dot{q}_{1\to 2} &= -a_1 \nabla T. \end{aligned} \tag{4.17}$$

Durch Einsetzen in die Wärmeleitungsgleichung und numerische Integration folgt dann der Finite-Differenzen-Stern für die Bauteilkopplung:

$$\begin{aligned} \frac{T^{t+1}_{x,y,z} - T^t_{x,y,z}}{\Delta t} &= a_2 \left(\frac{T^t_{x+1,y,z} - T^t_{x,y,z}}{\Delta x_1^2} \right) - a_1 \left(\frac{T^t_{x,y,z} - T^t_{x-1,y,z}}{\Delta x_1^2} \right) \\ &+ a_1 \left(\frac{T^t_{x,y+1,z} - 2T^t_{x,y,z} + T^t_{x,y-1,z}}{\Delta y_1^2} + \frac{T^t_{x,y,z+1} - 2T^t_{x,y,z} + T^t_{x,y,z-1}}{\Delta z_1^2} \right). \end{aligned} \tag{4.18}$$

Für diesen Ausdruck wird ein Nachbarpunkt aus dem groben Gitter benötigt, der durch zwei Interpolationen gewonnen werden kann (vgl. Abbildung 4.13 Schritt 2). Zu Beginn wird eine Interpolation des Temperaturwerts auf der y-z-Ebene aus vier Nachbarknoten mit Hilfe einer bilinearen Ansatzfunktion durchgeführt. Die verwendete Interpolationsvorschrift ist in Abbildung 4.11 dargestellt. Aus dem so erhaltenen Temperaturwert und dem Wert am Interface kann dann der benötigte Nachbarknoten (in einem Abstand dx_1 vom Interface) durch lineare Interpolation entlang der x-Achse berechnet werden. Auf den Randknoten des Interfaces werden die Randbedingungen unverändert aus Abschnitt 4.2.2 ausgeführt.



$$\begin{aligned} T(x,y) = T_1\ &(1-y)(1-z) \\ &+ T_2\ (1-z)\,y \\ &+ T_3\ z\,y \\ &+ T_4\ z(1-y) \end{aligned}$$

Abbildung 4.11: *Interpolationsvorschrift für die Bauteilkopplung*

Im letzten Schritt werden die zuvor berechneten Temperaturwerte vom Interface an die jeweiligen Gitterknoten der Bauteile kopiert. Für die Übertragung der Werte an das grobe Gitter müssen hierbei Interpolationen durchgeführt werden. Hierzu wird eine Interpolation aus vier Nachbarknoten mit Hilfe der in Abbildung 4.11 dargestellten bilinearen Ansatzfunktion verwendet.

Für nicht lineare Temperaturverläufe, z.B. bei der Betrachtung von Quellen oder Senken im Gebiet, ist ein höherer Differenzen-Ansatz am Interface notwendig. Hierzu muss am Bauteilinterface ein FDM-Ausdruck (für Gleichung 4.18) mit mehr als einem Nachbarpunkt in

jede Richtung verwendet werden. Außerdem werden dann für die verwendeten bilinearen Interpolationen Ansätze höherer Ordnung erforderlich. Dieses Verfahren wird im Rahmen dieser Arbeit nicht weiter betrachtet.

Bislang wurde von einer konstanten Zeitschrittweite auf allen Bauteilgittern ausgegangen. Für unterschiedliche räumliche Gitterauflösungen auf den Bauteilen ergeben sich jedoch, resultierend aus der Stabilitätsbedingung der FD-Methode (vgl. Gleichung 4.13), maximal mögliche Zeitschrittweiten Δt. Somit wird zwischen den Bauteilen eine zeitliche Synchronisation notwendig, bei der wie in Abbildung 4.12 dargestellt, die Knoten aus dem zurückliegenden Zeitschritt zur Kopplung an den Kontaktflächen verwendet werden. Bei stark

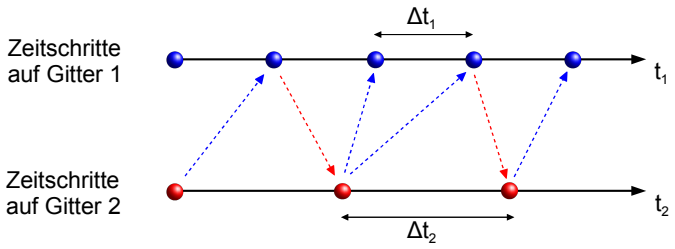


Abbildung 4.12: *Kopplung für unterschiedliche Zeitschrittweiten*

von einander abweichenden Zeitschrittweiten ist eine lineare Extrapolation der Zeitschritte denkbar, die im Rahmen dieser Arbeit allerdings nicht weiter betrachtet wurde. Die Validierung der Bauteilkopplung für einen mehrschichtigen Wandaufbau wird in Abschnitt 6.1.4 gegeben.

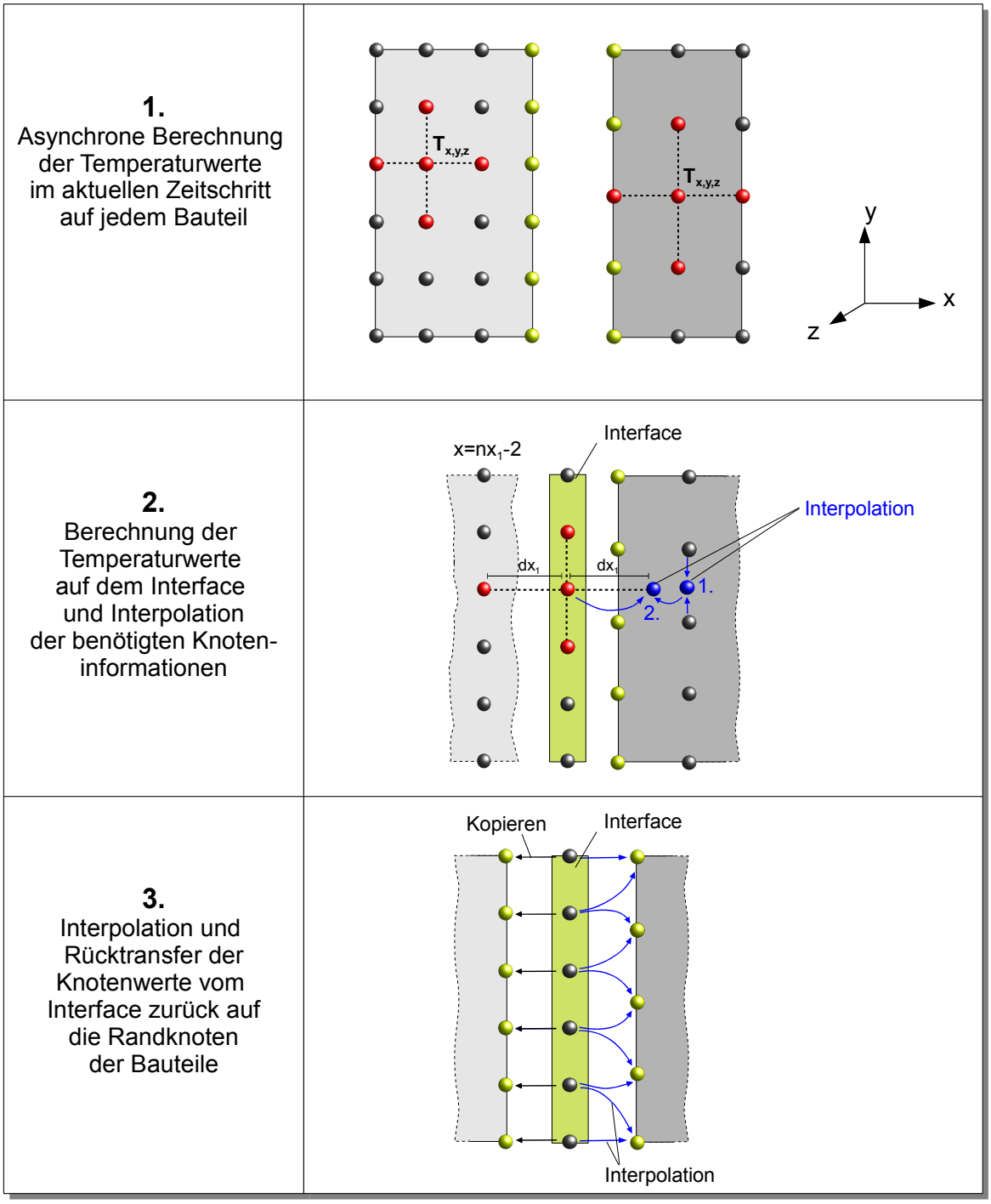


Abbildung 4.13: *Bauteilkopplung: Ablauf der Berechnung in jedem Zeitschritt*

4.2.4 Verteilte Berechnung auf Grafikkarten mit CUDA

In diesem Abschnitt wird ein effizienter hardwarebeschleunigter FD-Ansatz zur schnellen Berechnung des Wärmeleitungsproblems auf Grafikkarten (GPUs) vorgestellt. Die Nutzung von Grafikprozessoren für allgemeine wissenschaftliche Berechnungen, auch als GPGPU (General Purpose Computing on GPUs) bezeichnet, ist erst in den letzten Jahren aufgekommen und wurde durch die Entwicklung neuer Programmierschnittstellen wie Nvidias CUDA-Framework [84], dem OpenCL Standard der Khronos Group [66] und AMDs ATI Stream Technologie [4] stark voran getrieben. Grafikkarten haben sich zu leistungsstarken massiv-parallelen Mehrkernsystemen mit deutlich höherer Rechenleistung und Speicherbandbreite als CPUs entwickelt. Als Beispiel sei hier die Geforce GTX 480 Grafikkarte von Nvidia aufgeführt, die mit 480 Recheneinheiten und einer Speicherbandbreite von ca. 170 GB/s, theoretisch über 1300 GFLOPS/s (Milliarden Gleitkommaoperationen pro Sekunde) erreicht. Im Vergleich hierzu erlangt ein aktueller Core i7 Prozessor mit 4 Recheneinheiten und ca. 30 GB/s Speicherbandbreite gerade einmal 50 GFLOPS/s. Der Grund für diesen deutlichen Geschwindigkeitsunterschied zwischen GPU und CPU liegt in der für berechnungsintensive massiv-parallele Aufgaben optimierten Hardwarearchitektur der GPUs. Hierbei wird ein Großteil der Transistoren des Chips für Recheneinheiten (ALUs Arithmetic logic units) verwendet. Puffer-Speicher (Cache) und Steuerungseinheit sind im Vergleich zur CPU klein (vgl. Abbildung 4.14). Grafikkarten führen eine Berechnung gleichzeitig (parallel) auf vielen Daten aus (SIMD). Wenn viele gleichartige Daten auf gleiche Weise bearbeitet werden sollen (z.B. bei Matrizenoperationen oder beim Rendering) sind Grafikkarten normalen CPUs, die deutlich weniger Daten parallel bearbeiten können, weit überlegen.

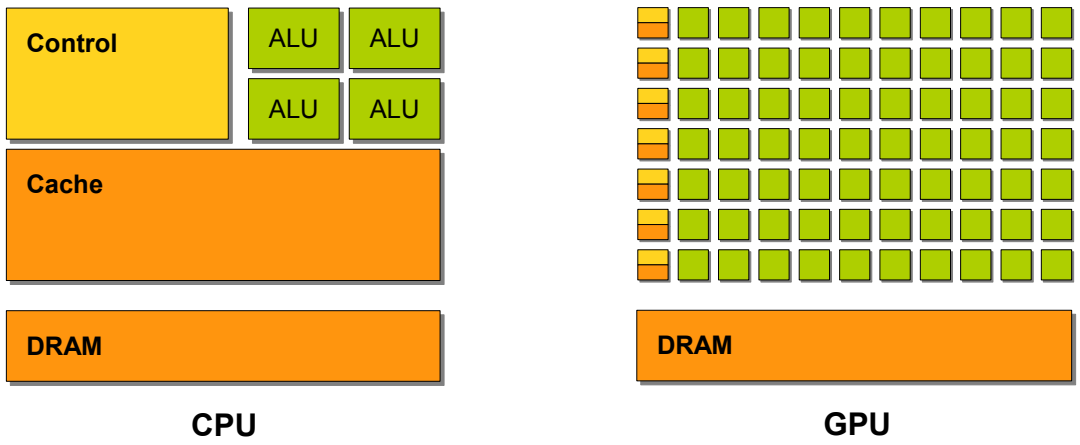


Abbildung 4.14: *Unterschiedliche Hardwarearchitektur von CPU und GPU [85]*

Eine effiziente Implementierung zur Lösung des Wärmeleitungsproblem auf Grafikkarten unter Verwendung der Compute Unified Device Architecture von Nvidia wird im Folgenden erläutert. Hierdurch lassen sich komplexe thermische Simulationen um bis zu 50-mal schneller als auf einer Standard Single-Core-CPU lösen.

Die CUDA Architektur

CUDA ist eine von Nvidia entwickelte Technik die es ermöglicht, bestimmte rechenintensive Programmteile auf der Grafikkarte signifikant schneller abzuarbeiten. Die grundlegende CUDA Architektur wird im folgenden kurz erläutert. Eine typische CUDA-fähige Grafikkarte ist unterteilt in bis zu 60 Streaming Multiprozessoren (SMs) (GTX 480) [84, 85, 107]. Zwei solcher SMs bilden einen sogenannten Block und teilen sich einen gemeinsamen Datencache und eine Steuerungseinheit. Jeder Streaming Multiprozessor hat in der Regel 8 Streaming Prozessoren (SPs) mit je einer Multiplikationseinheit und einer Additionseinheit. Auf den massiv multi-threading fähigen Streaming Multiprozessoren können bis zu 1024 Threads gleichzeitig laufen (also über 60000 Threads auf der GTX 480 mit 60 solcher SMs). Die Struktur einer CUDA-fähigen Grafikkarte ist in Abbildung 4.15 dargestellt.

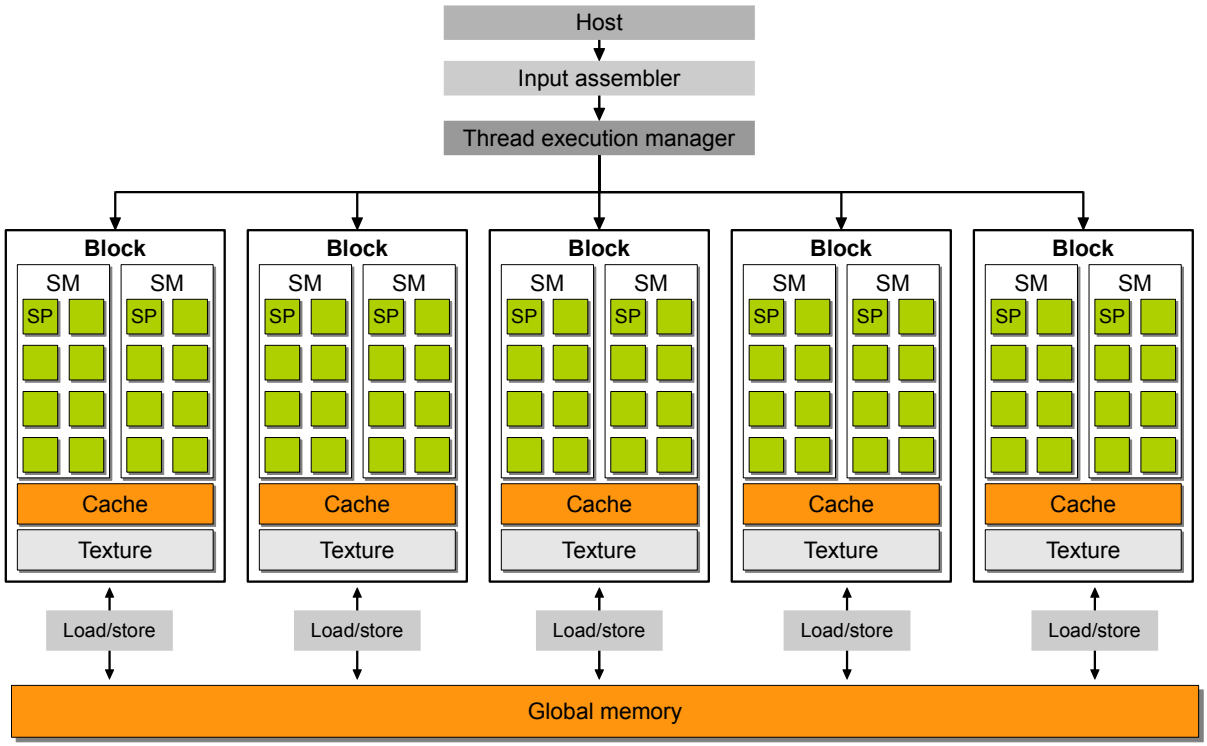


Abbildung 4.15: *Struktur einer CUDA-fähigen Grafikkarte [67]*

Die Hauptarbeit bei der Programmierung mit CUDA, wie bei der Multi-Core-Programmierung auch, besteht in der effizienten Aufteilung der Last (Workload) auf einzelne Threads. Der Unterschied liegt lediglich in der Anzahl der parallel ausführbaren Threads, diese ist auf CPUs häufig auf vier begrenzt und kann auf GPUs mehrere 100 betragen. Auf den Threads läuft grundsätzlich immer der selben Code, Nvidia nennt das SIMT-Technik (Single Instruction Multiple Threads). Dieser Thread-Code, der Kernel, sollte klein sein und möglichst wenige Verzweigungen aufweisen. Die Threads werden zu Blöcken zusammengefasst und mehrere Blöcke bilden dann ein sogenanntes Grid. Die Blöcke mit

maximal 1024 Threads werden dann einer der 60 Streaming Multiprozessoren (GTX 480) zugewiesen und hier auf die echten Hardware-Threads, in sogenannte Warps, verteilt. Warps fassen 32 Threads zusammen, die dann per Pipeline den Recheneinheiten (SPs) zugeführt werden. Den maximal möglichen 61440 ($60 \cdot 1024$) stehen nur 480 physische Recheneinheiten gegenüber. Somit sind immer ausreichend wartende Threads vorhanden, um die Recheneinheiten auszulasten.

Ein CUDA Programm besteht aus zwei Bestandteilen: dem auf dem Host (CPU) ausgeführten Code und dem auf dem Device (GPU) ausgeführten Code. Der sequenzielle Teil der Anwendung läuft auf dem Host und der rechenintensive Teil auf dem Grafikprozessor. Mit der CUDA API stellt Nvidia eine auf ANSI C basierende Schnittstelle zur Verfügung, um Host- und Device-Code zu entwickeln.

Eine detaillierte Beschreibungen der CUDA Architektur und Programmierparadigmen kann u.a. Kirk et al. [67], Sanders et al. [99] sowie dem aktuellen CUDA Programming Guide von Nvidia [85] entnommen werden.

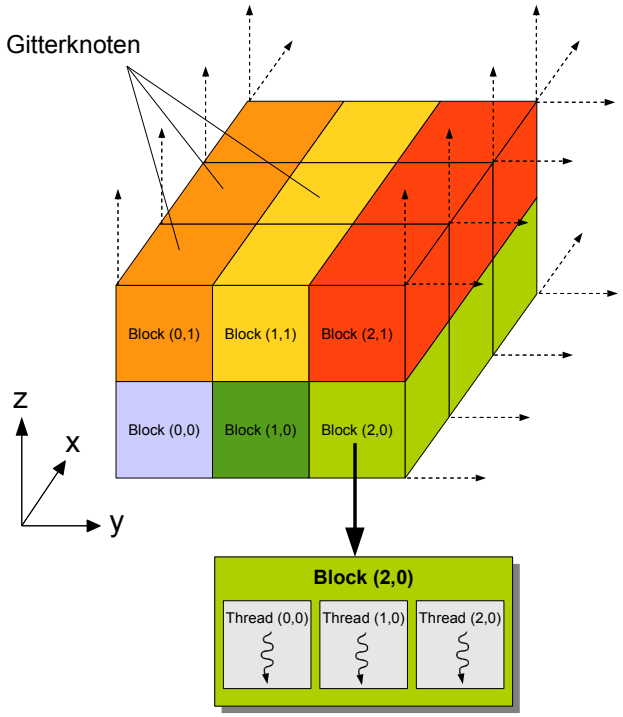


Abbildung 4.16: *Gitter von Threadblöcken*

Eine CUDA 3D FDM Implementierung

Zur hardwarebeschleunigten Simulation der Wärmeleitung wurde ein Ansatz auf Basis von CUDA entwickelt. Hierbei wird für die Berechnung des neuen Temperaturwertes an jedem Knoten (auf dem Berechnungsgitter) ein CUDA-Thread verwendet. Insgesamt werden für

die Berechnung in jedem Zeitschritt auf einem Berechnungsgitter (mit nx, ny und nz Knoten in die jeweilige Richtung) $nx \cdot ny \cdot nz$ Threads auf der GPU ausgeführt, die von den Recheneinheiten parallel bearbeitet werden. Das 3D Rechengitter ist in einem 1D Array abgelegt, so dass die Daten kontinuierlich im Speicher liegen und erlauben somit einen effizienten Zugriff. Die Knoten bzw. die darauf arbeitenden Threads sind abhängig von der Topologie des Gebiets in sogenannte Blöcke eingeteilt. Hierzu ist das Gebiet entlang der yz-Ebene Knotenweise unterteilt (vgl. Abbildung 4.16). Jedem Block sind die Knoten in x-Richtung zugeordnet. Ein Block wird von zwei (je nach Architektur) Streaming Multiprozessoren bearbeitet, die sich einen gemeinsamen Speicher (shared memory) teilen (vgl. Abbildung 4.15).

Liegen Nachbarknoten, die für den Finite-Differenzen-Stern benötigt werden, außerhalb des Blocks (also nicht im selben Cache) wird der Zugriff über den Hauptspeicher der GPU ausgeführt. Dieser ist deutlich langsamer als der Cache. Aus diesem Grund wurde eine Variante des Kernels entwickelt, der den Texturspeicher (Texture Memory) der Grafikkarte verwendet. Der Texturspeicher ist ein besonderer Speicher, der ursprünglich für das Rendern von Szenen mit OpenGL und DirectX verwendet wurde, aber aufgrund seiner Eigenschaften auch für allgemeine Berechnungen gut geeignet ist. Bei Verfahren wie der Finite-Differenzen-Methode mit Speicherzugriffen mit hoher räumlicher Lokalität ist die Verwendung des Texturspeichers durch seine höhere Bandbreite und die vom Chip reduzierten Zugriffe auf den GPU Hauptspeicher sehr effizient.

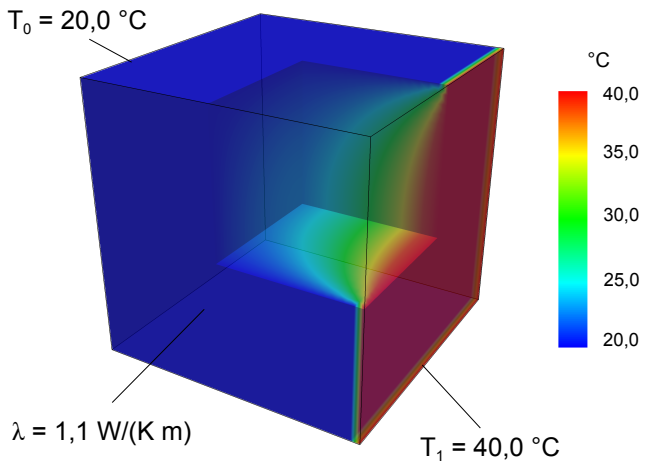


Abbildung 4.17: *Temperaturverteilung in Quader*

Im Folgenden zeigt ein Benchmark die Effizienz des implementierten Ansatzes. Hierzu wurde ein Quader mit einer Gitterauflösung von 128^3 betrachtet, der mit konstanten Temperaturwerten am Rand versehen wurde (vgl. Abbildung 4.17). Für die Simulation wurden 10.000 Iterationsschritte auf vier verschiedenen CUDA-fähigen Grafikkarten berechnet. In Abbildung 4.18 ist der Speedup (Beschleunigung) im Verhältnis zur Berechnung auf einer Single-Core Xeon(R) CPU mit 2.13 GHz für zwei Varianten (jeweils unter Nutzung des GPU Hauptspeichers und des Texturspeichers) dargestellt. Zu Erkennen ist, dass sich unter Verwendung

des Texturspeichers die Effizienz des Verfahrens fast verdoppelt. Auf Highend Grafikkarten, wie der Tesla C1060, konnte ein Speedup von ca. 45 erreicht werden, aber bereits auf Karten im unteren Preissegment werden noch Beschleunigungen von bis zu 20 erreicht.

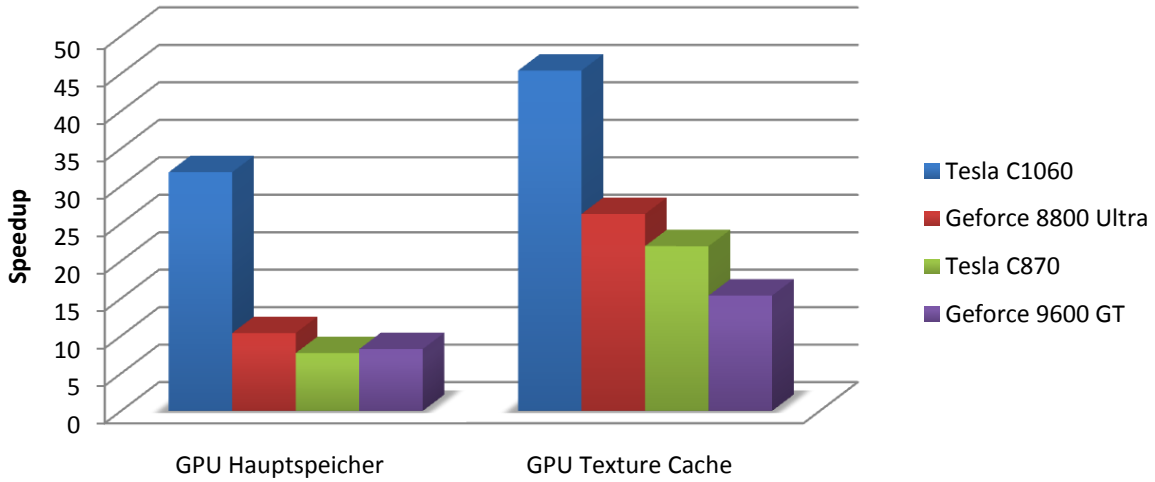


Abbildung 4.18: *Speedup im Verhältniss zur Berechnung auf einer Xeon(R) CPU mit 2.13 GHz*

5 Eine interaktive Simulationsumgebung

Eine optimale Gebäudeplanung kann nur unter Berücksichtigung der zu erwartenden thermischen Verhältnisse im und am Bauwerk erfolgen. Erst durch detaillierte Kenntnisse dieser Verhältnisse lassen sich energieeffiziente und nachhaltige Gebäude planen. Die komplexen zeitabhängigen thermischen Prozesse können häufig nur mit Hilfe von Computersimulationen erlangt werden. Typischerweise haben solche Feldmodell-basierten Simulationen (im Gegensatz zu Zonenmodellansätzen wie sie z.B. von Trnsys [113] zur Gebäudesimulation verwendet werden) lange Laufzeiten und der Benutzer kann nicht in die laufende Berechnung eingreifen. Der Ablauf gleicht hierbei einer Art Stapelverarbeitung; zu Beginn wird ein Setup (Geometrie + Randbedingungen) vom Planer erstellt, dann wird die Simulation gestartet. Je nach Komplexität der Problemstellung kann die Berechnung mehrere Tage bis Wochen dauern, erst nach Beendung können die Ergebnisse analysiert werden. Wird dann bei der Auswertung festgestellt, dass z.B. im Rahmen einer Optimierung Geometrie oder Randbedingungen angepasst werden sollen, so muss der gesamte Prozess wiederholt werden. Dieser starre Ablauf, der in den meisten kommerziellen Simulationsprogrammen zu finden ist, macht die Einbindung von Computersimulationen in die Planungsprozesse von Gebäuden schwierig. Insbesondere für die kooperative Optimierung eines Bauwerks durch eine Gruppe von Fachplanern ist dieser Ansatz nicht geeignet.

Im Rahmen dieser Arbeit wurde eine interaktive Simulationsumgebung entwickelt, die unter Nutzung effizienter paralleler numerischer Verfahren (vgl. Kapitel 4) thermische Simulationen mit kurzen Antwortzeiten erlaubt, die direkt visualisiert werden. Hierbei kann der Benutzer interaktiv, d.h. zur Laufzeit der Simulation, die Ergebnisse analysieren und in das System eingreifen, um beispielsweise Änderungen an der Gebäudegeometrie vorzunehmen. Die Simulationsumgebung erlaubt einer Gruppe von Fachplanern interaktiv Gebäude unter thermischen Gesichtspunkten zu optimieren und in den Gebäudeplanungsprozess einzubeziehen. Die interaktive Steuerung von Computersimulationen (sogenannte Computational Steering-basierte Simulationen) wird im Abschnitt 5.1 erläutert.

Als Grundlage zur Konstruktion von Gebäudemodellen dient AutoCAD Architecture, das direkt an die Simulation gekoppelt ist. Geometrien, Materialeigenschaften und weitere Randbedingungen werden über die CAD-Software vorgegeben und an die laufende Simulation übertragen. Durch Einsatz eines CAD-basierten Entwurfsraumes können geometrisch sehr komplexe Problemstellungen nicht nur transient simuliert werden, sondern es ist auch möglich, das Systemverhalten interaktiv zu optimieren. Der entwickelte Entwurfsraum auf Basis von AutoCAD wird in Abschnitt 5.2 vorgestellt.

Der Computational Steering Ansatz stellt durch die Integration von komplexer Simulation, Datenanalyse, Visualisierung und Nachbearbeitung sehr hohe Anforderungen an die unterstützende Hardware und Algorithmik [90, 124]. Aus diesem Grund wird neben der parallelen Simulation auch die Visualisierung verteilt auf mehreren Grafikkarten durchgeführt. Der entwickelte Ansatz, basierend auf CGLX der Cross-Platform Cluster Graphic Library, ermöglicht eine effiziente Darstellung großer Datenmengen, die in der thermischen Simulation in jedem Zeitschritt anfallen. Die CGLX-Bibliothek ist ein OpenGL Framework für skalierbare verteilte Visualisierungsumgebungen, das es ermöglicht Daten auf Tiled-Display-Umgebungen mit sehr hohen Auflösungen (>300 Megapixel) zu rendern [36]. Eine Tiled-Display-Umgebung ist eine Bildschirmwand bestehend aus vielen einzelnen zusammengeschaltet Displays, welche in der Regel von einem Rendercluster angesteuert werden. Der verteilte Renderansatz erlaubt eine schnelle Ausgabe und Manipulation sehr großer Datenmengen [31]. Für kooperative Planungsprozesse haben sich solche Tiled-Display-Umgebungen als besonders vorteilhaft erwiesen [2, 72]. Eine Gruppe von Planern kann hierbei mit dem Gebäudemodell virtuell interagieren, sich um das Modell bewegen und es von unterschiedlichen Standpunkten betrachten sowie das Modell berühren. Des Weiteren lassen sich sich auf dem Tiled-Display auch weitere Planungsdaten (wie Konstruktionspläne oder Grafiken) neben der Simulation darstellen. Somit wird eine kooperative Gebäudeoptimierung möglich, die mit konventionellen Computerbildschirmen nicht zu erreichen wäre. Der Ansatz zur verteilten Visualisierung und Darstellung auf Tiled-Display-Systemen wird in Abschnitt 5.3 erläutert. Abbildung 5.1 zeigt den generellen Systementwurf der interaktiven Simulationsumgebung bestehend aus vier Komponenten: dem Modellierer auf Basis von AutoCAD, einem Steuerungsrechner zur Synchronisation der Simulation und Visualisierung, dem verteilten thermischen Simulationskern sowie der verteilten Visualisierung mit Ausgabe auf dem Tiled-Display-System.

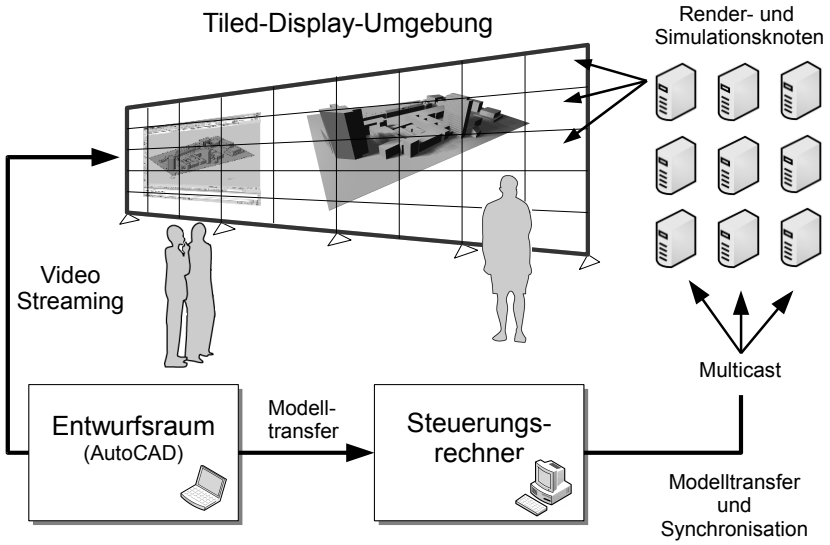


Abbildung 5.1: *Systementwurf der interaktiven Simulationsumgebung*

5.1 Computational Steering

Der klassische Ablauf einer wissenschaftlichen Berechnung verläuft in einer Art Stapelverarbeitung, bei der die einzelnen Teilschritte sequentiell nacheinander bearbeitet werden. Zu Beginn steht der Pre-processing Schritt, in dem die Geometrie aufbereitet wird und Randbedingungen festgelegt werden. Anschließend folgt die Berechnung, nach deren Ablauf die Ergebnisse in einem sogenannten Post-processing Schritt visualisiert und analysiert werden (vgl. Abbildung 5.2a). Dieser Ablauf führt zu einer Entkopplung von Simulation und Analyse und macht ein Steuern der Berechnung zur Laufzeit unmöglich.

Sollen komplexe Computersimulationen im Rahmen der Gebäudeplanung genutzt werden, so sind Simulationswerkzeuge notwendig, die das Durchführen von Fallstudien innerhalb kurzer Zeit erlauben. Hierbei müssen die Fachplaner und Ingenieure in der Lage sein den zeitlichen Verlauf einer Simulation zu verfolgen, um ein Verständnis über die ablaufenden, zumeist komplexen physikalischen Prozesse zu erhalten. Des Weiteren muss ein Eingreifen in die Simulation zur Laufzeit möglich sein, um beispielsweise Änderungen an der Geometrie oder den Randbedingungen vorzunehmen. Dieser Ansatz von Visualisierung und gleichzeitiger interaktiver Steuerung wird als Computational Steering bezeichnet [81] (vgl. Abbildung 5.2b).

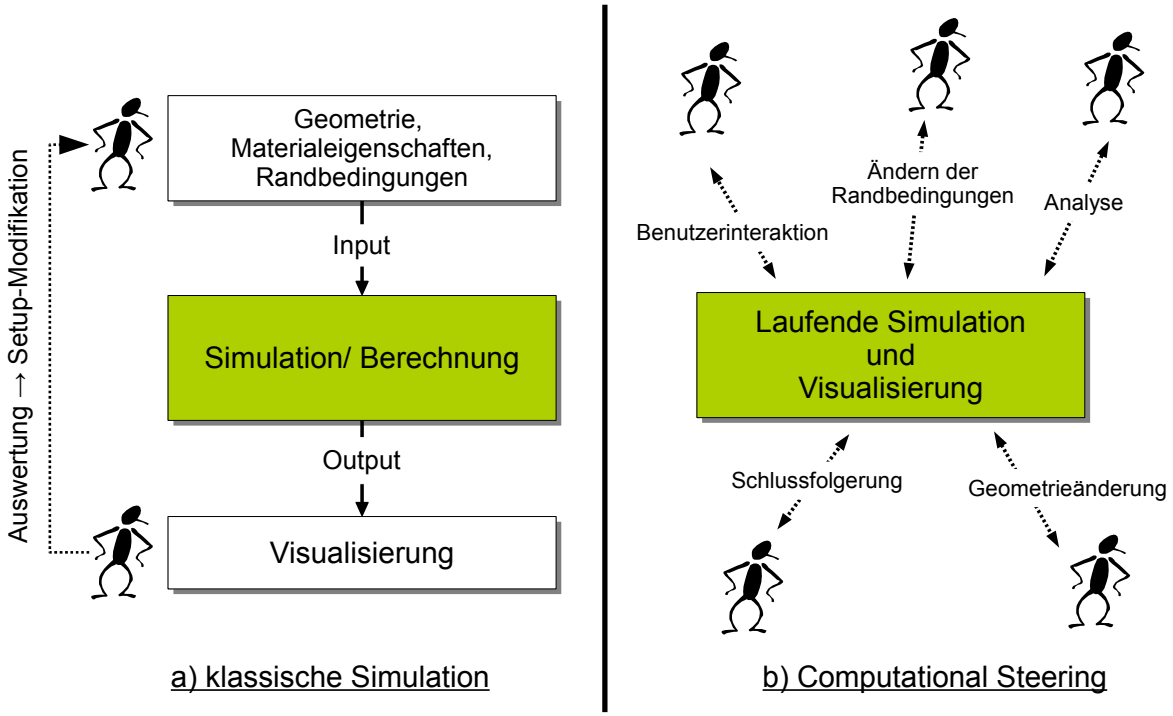


Abbildung 5.2: *Vergleich zwischen klassischer Simulation und Computational Steering basierter Simulation*

Ziel des Computational Steering Ansatzes ist es, durch direkte Modifikation des Simulationssystems zur Laufzeit, d.h. ohne Unterbrechung der laufenden Simulation, ein schnelles und detaillierteres Wissen der Auswirkungen von Änderungen zu erlangen. Hierbei werden geometrische Modellierung (Pre-processing), Simulation, Datenaufbereitung, Post-processing und Visualisierung gleichzeitig innerhalb einer Anwendung ausgeführt, so dass eine interaktive Analyse der Daten erst möglich wird [22, 23, 74, 90, 112, 123, 124]. Die Vorteile von Computational Steering basierten Simulationsumgebungen sind im Folgenden aufgelistet [65]:

- Erhöhte Produktivität aufgrund schnellerer Durchführung der Simulationen.
- Ideen und Vorschläge können umgehend demonstriert und analysiert werden.
- Zusammenhänge zwischen Ursache und Wirkung werden schneller ersichtlich.

Im Rahmen dieser Arbeit wurde eine Computational Steering basierte Simulationsumgebung entwickelt, welche die Visualisierung und gleichzeitige interaktive Steuerung von thermischen Simulationen ermöglicht. Die Hauptbestandteile der Anwendung werden in den folgenden Abschnitten erläutert.

5.2 Ein virtueller Entwurfsraum auf Basis von AutoCAD

Der Entwurf und die Konstruktion von Gebäuden erfolgt heutzutage rechnergestützt durch CAD-Programme (Computer-Aided-Design). Hierbei werden virtuelle (meist dreidimensionale) Gebäudemodelle erstellt, aus denen die benötigten Planungsdokumente wie Schnitte, Ansichten und Grundrisse abgeleitet werden können. Vorteile der computergestützten Konstruktion sind neben dem einfachen elektronischen Austausch von Planungsdaten, die Umsetzung unterschiedlicher konstruktiver Varianten eines Bauwerks sowie das einfache Einbringen von Änderungen während des Planungsprozesses. Des Weiteren ist es möglich, neben der Geometrie eines Bauteils zusätzliche Informationen, wie die physikalischen Eigenschaften oder den Bauablauf zu speichern. Vor diesem Hintergrund wurde das entwickelte thermische Simulationsmodell in einen interaktiven virtuellen Entwurfsraum auf Basis eines CAD-Systems eingebettet. Somit können Geometrien und Eigenschaften vom Fachplaner über die CAD-Anwendung vorgegeben werden, die als Vorgabe für die Simulation dienen (vgl. Abbildung 5.3).

Grundlage des entwickelten Konstruktionsraums ist das CAD-Programm AutoCAD Architecture der Firma Autodesk [7], welches um notwendige Funktionalitäten erweitert wurde. AutoCAD ist das weltweit am meisten verbreitete CAD-Produkt auf dem Markt mit mehr als 3 Millionen Lizenzen. Es bietet Werkzeuge zur Erstellung von zwei- und dreidimensionalen Modellen und wird überdisziplinär in mehreren konstruktiven Bereichen wie dem Maschinenbau, der Elektrotechnik und dem Bauwesen genutzt. Als bauspezifische Erweiterung wird von der Firma Autodesk AutoCAD Architecture angeboten, das eine komplette (IFC-konforme) Gebäudeplanung ermöglicht. Über die objektorientierte Programmierschnitt-

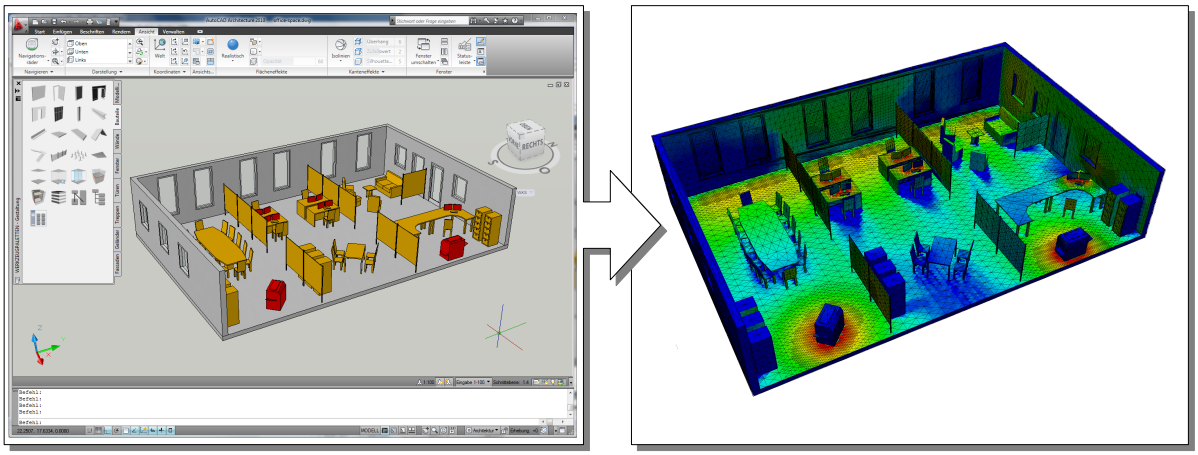


Abbildung 5.3: *Kopplung vom CAD-basiertem Entwurfsraum an das Simulationsmodell*

stelle OMF (Object Modeling Framework) [5] konnte AutoCAD u.a. um Module zum Preprocessing, für den Datenaustausch, zur Ereigniserkennung und Grenzwertüberwachung, sowie zur Aufbereitung des Datenmodells ergänzt werden.

Im Folgenden wird AutoCAD Architecture und seine Programmierschnittstelle OMF erläutert. Außerdem werden die entwickelten AutoCAD Erweiterungen zur virtuellen Gebäudeplanung unter Einbeziehung thermischer Simulationen vorgestellt.

5.2.1 Allgemeines zu AutoCAD Architecture

AutoCAD Architecture (früher: Architectural Desktop (ADT)) der Firma Autodesk [7] ist eine Erweiterung von AutoCAD für den Bau- und Architekturmarkt. Die AutoCAD Grundfunktionalitäten zur zwei- und dreidimensionalen Konstruktion sind in AutoCAD Architecture um bauspezifische Module ergänzt, welche die Planung des Bauwerks über die einzelnen Planungsphasen (Konzeptuelle Planung - Entwurf - Konstruktionsausarbeitung) bis hin zur Gebäudenutzung ermöglichen. AutoCAD Architecture stellt ein objektorientiertes Gebäudemodell auf Basis der Industry Foundation Classes (vgl. Abschnitt 3.1) bereit. Hierzu wird eine umfangreiche Bibliothek von Bauelementen (wie Wände, Stützen, Träger, Fenster und Türen) für eine komponentenorientierte Konstruktion zur Verfügung gestellt (vgl. Abbildung 5.4).

Jedes Bauteil wird über einen sogenannten Bauteilstil definiert und besteht aus seinen physikalischen Eigenschaften (wie Höhe, Breite, Form und Lage). Das Bauteil besitzt keine Informationen über seine grafische Darstellung, diese werden über eigenständige Objekte zur Repräsentation vorgegeben. Der Vorteil in der strikten Trennung zwischen physikalischen Eigenschaften und grafischer Darstellung liegt in der Möglichkeit, unterschiedliche Ansichten der Bauteile zu realisieren. Beispielsweise kann in einem Grundrissplan für die Darstellung eines Bauteils ein anderes Symbol verwendet werden als in einer isometrischen Darstellung.

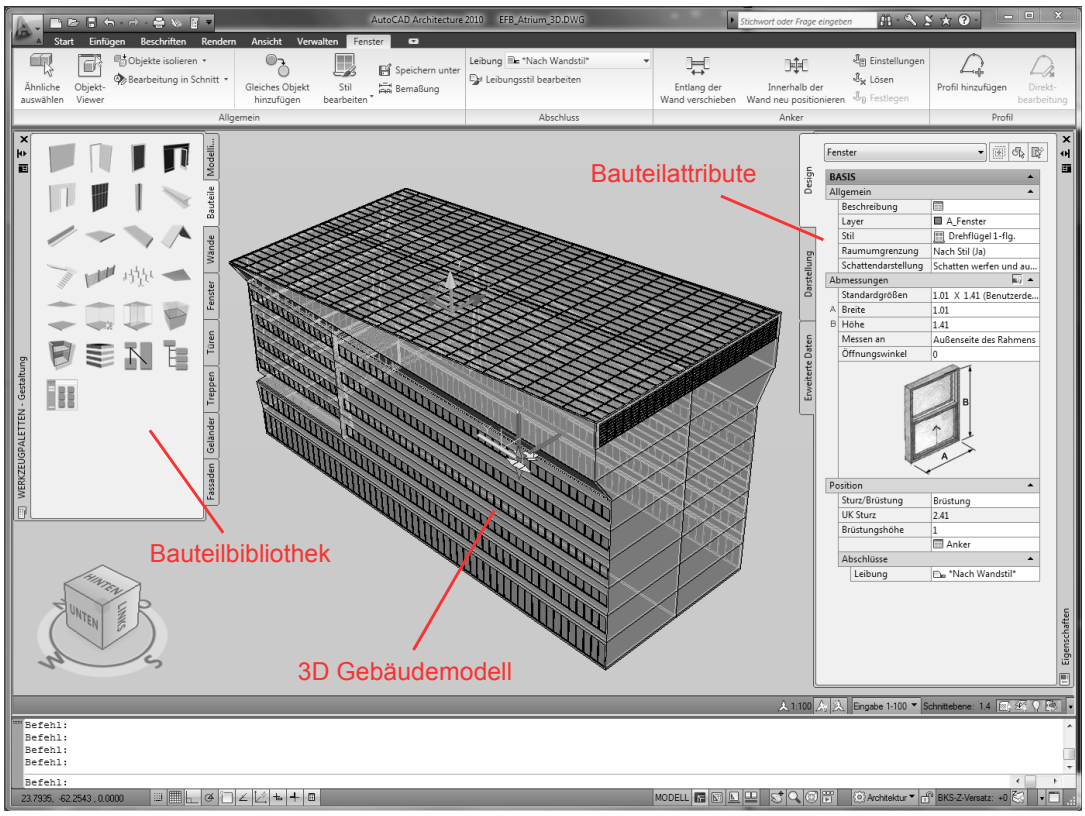


Abbildung 5.4: *Gebäudemodellierung in AutoCAD Architecture 2010*

Alle für den Bau notwendigen Planungsdokumente (wie Grundrisse, Ansichten und Schnitte) können somit aus dem 3D-Modell automatisch erstellt werden. Des Weiteren stellt AutoCAD Architecture konstruktionsstützende Maßnahmen zur Verfügung, so wird jedes Bauteil mit einer Positionskontrolle eingefügt, beispielsweise werden beim Verändern der Position einer Wand angrenzende Wände und Räume automatisch mitgeführt.

AutoCAD Architecture stellt eine ideale Plattform zur produktmodellbasierten Planung dar, mit dem sich komplexe Bauwerke vollständig digital abbilden lassen [83]. Außerdem stellt AutoCAD leistungsfähige Programmierschnittstellen zur Verfügung, die es ermöglichen, das System um eigene Funktionalitäten zu erweitern und Abläufe zu automatisieren. Diese werden im Folgenden Abschnitt dargestellt.

5.2.2 Die Programmierschnittstellen von AutoCAD

AutoCAD bietet eine Vielzahl von Programmierschnittstellen, die sich grob in zwei Gruppen einteilen lassen. Zur ersten Gruppe zählen u.a. die Programmiersprachen AutoLISP und Visual Basic for Applications (VBA), mit denen sich einfache Abläufe automatisieren lassen oder Zeichnungsobjekte verändert werden können. Die zweite Gruppe mit den Sprachen ObjectARX und OMF bilden die Grundlage auf der AutoCAD selbst aufgebaut ist und ermög-

lichen Zugriff auf das gesamte System. Hierdurch lassen sich tiefgehende Änderungen an der AutoCAD-Plattform vornehmen und stellen somit eine ideale Schnittstelle zur Entwicklung der benötigten Funktionalitäten des virtuellen Entwurfsraums dar.

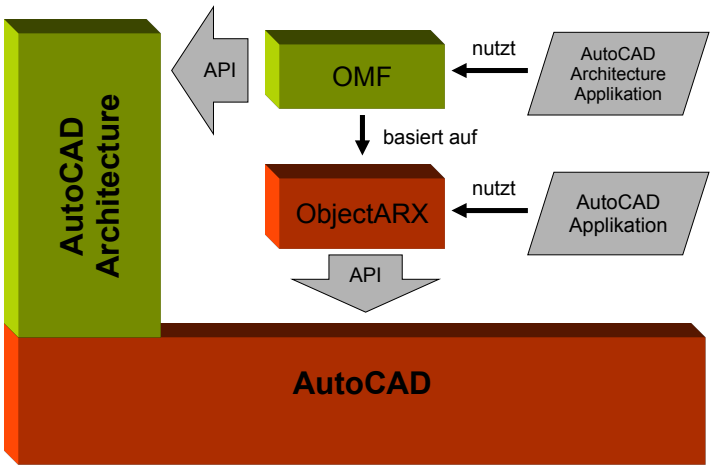


Abbildung 5.5: *Aufbau von AutoCAD und AutoCAD Architecture*

Das Object Modeling Framework ist eine Erweiterung von ObjectARX für Gebäudemodelle und ist Grundlage von AutoCAD Architecture. Reine AutoCAD Anwendungen werden hingegen mit ObjectARX entwickelt (vgl. Abbildung 5.5). Eine detaillierte Beschreibung kann McAuley [77], Kramer [70] sowie den ARX und OMF Entwicklerhandbüchern [5, 6] entnommen werden.

Das ObjectARX - Framework

ObjectARX (**A**utoCAD **R**untime e**X**tension) ist eine Programmierschnittstelle auf Basis von C++ für AutoCAD. Sie ermöglicht die Entwicklung von Applikationen, die den Funktionsumfang des CAD-Systems erweitern [6]. Da ObjectARX die Grundlage bildet, auf der die AutoCAD-Plattform entwickelt wurde, ist ein Zugriff auf nahezu alle Komponenten des Systems möglich. So lässt sich u.a. die interne Objektdatenbank um neue geometrische Objekte ergänzen, die Programmoberfläche an eigene Anforderungen anpassen sowie auf alle Zeichnungselemente und ihre Eigenschaften zugreifen. Zur Entwicklung eigener Erweiterungen stellt Autodesk eine objektorientierte Schnittstelle bestehend aus zahlreichen Klassen zur Verfügung, die zur besseren Übersicht in fünf Bibliotheken gegliedert sind (vgl. Tabelle 5.1).

ObjectARX Applikationen sind dynamische Bibliotheken (auch dynamic link library (dll) genannt), die zur Laufzeit des Hauptprogramms nachgeladen werden. Viele Grundfunktionen von AutoCAD sind in solche Bibliotheken ausgelagert und werden erst bei Bedarf vom Programm geladen. Durch den modularen Aufbau ist es ausreichend, beim Ändern oder Hinzufügen von Funktionalitäten nur diese dll neu zu übersetzen und nicht das gesamte Programm. Beim Linken gegen diese Bibliothek wird der *acrxEntryPoint()* als interner Einstiegs-

AcRx	Klassen zum Binden der Applikation und zur Registrierung und Identifikation in der Laufzeitumgebung
AcEd	Klassen zur Registrierung einfacher AutoCAD-Befehle
AcDb	AutoCAD Datenbank-Klassen
AcGi	Grafik Klassen zum Rendern von AutoCAD-Objekten
AcGe	Allgemeine Klassen für Geometrie-Objekte

Tabelle 5.1: *ObjectARX-Bibliotheken*

punkt für AutoCAD festgelegt. Über diese Funktion werden alle Nachrichten zwischen AutoCAD und der dll verarbeitet. Sie wird beim Öffnen und Schließen der Zeichnung und beim Laden und Entladen des Moduls aufgerufen. Das Modul kann somit auf sämtliche Interaktionen in AutoCAD reagieren. Die ObjectARX-dlls haben die Endung .arx oder .dbx und teilen den selben Adressbereich wie AutoCAD.

ObjectARX ermöglicht einen effizienten Zugriff auf die Zeichnungselemente, die Programmoberfläche sowie die Laufzeitumgebung und ist optimal geeignet, um Abläufe in AutoCAD zu automatisieren und neue Funktionen zu implementieren.

Das Object Modeling Framework (OMF)

Das Object Modeling Framework (OMF) [5] ist eine Erweiterung von ObjectARX für die Entwicklung von AutoCAD Architecture Zusatzapplikationen (vgl. Abbildung 5.5). OMF stellt ein Gebäudemodell für den AEC-Bereich (Architecture, Engineering and Construction) zur Verfügung, so dass intelligente Bauteile wie Wände, Türen und Fenster über diverse Anwendungen hinweg genutzt werden können. Es ist möglich neue Bauteile über OMF zu definieren, sowie bestehende zu erweitern. Diese Objekte sind weiterhin kompatibel zu dem AutoCAD (DWG) Dateiformat. Für den Zugriff auf das bauteilorientierte Gebäudemodell stellt OMF eine umfangreiche Klassenbibliothek zur Verfügung, über die eigene Funktionen und Objekte in das Datenmodell von AutoCAD Architecture integriert werden können.

In einer OMF-Anwendung können die Standard ObjectARX Funktionen verwendet werden. Erst für komplexe Anwendungen im Gebäudebereich sind OMF Methoden notwendig. Hierbei lohnt der relativ hohe Einarbeitungsaufwand in die OMF-API nur für Projekte, in denen mehrere grafische Darstellungen für Bauteile entwickelt oder neue stilbasierte Objekte mit Verbindungen zu anderen Objekten definiert werden sollen [5].

Das Object Modeling Framework basiert auf fünf Säulen, welche zu deutlichen Vorteilen gegenüber traditionellen CAD-Systemen führen:

1. **Verwendung eines vollständig objektorientierten Entwurfs**: Bauteile werden als eigenständige Objekte vorgehalten, an die Anfragen z.B. bzgl. ihres Volumens oder ihrer Fläche gestellt werden können oder wie das Bauteil in einer bestimmten Ansicht darzustellen ist.

2. **Bereitstellen eines allgemeinen Gebäudemodells**: Übergeordnete Zusammenfassung aller Gebäudeinformationen und Partialmodelle innerhalb eines einheitlichen virtuellen Gesamtmodells. Das Modell beinhaltet hierbei alle für die Gebäudeplanung und Bauausführung relevanten Informationen, zum Beispiel Geometriedaten, Informationen zum Design, Materialeigenschaften und Randbedingungen.
3. **Trennung der physikalischen Daten von der grafischen Darstellung eines Bauteils**: Jedes Bauteilobjekt in OMF ist unterteilt in physikalische Eigenschaften und grafische Darstellung. Für die Darstellung werden je nach Ansicht unterschiedliche Representationsobjekte verwendet. So ist es möglich, verschiedene Visualisierungen für ein Bauteil zu realisieren.
4. **Definieren von Bauteilbeziehungen**: Für Beziehungen (Verbindungen) zwischen Bauteilen werden in OMF sogenannte Ankerobjekte verwendet. So ist beispielsweise eine Wand über einen Anker mit einem auf ihr platzierten Fenster verknüpft. Wird das Fenster verschoben, passt sich die Aussparung in der Wand automatisch an.
5. **Interoperabilität der Anwendungen**: Alle Objektdefinitionen (Bauteilstile) sind über frei erhältliche DBX-Module in anderen Autodesk Programmen nutzbar. Somit lassen sich bauteilorientierte Gebäudemodelle auch unter anderen Autodesk Plattformen, die nicht wie AutoCAD Architecture aus dem AEC-Bereich stammen, nutzen.

5.2.3 Entwickelte Funktionen des Konstruktionsraums

Die hier vorgestellten Erweiterungen von AutoCAD Architecture basieren auf dem Object Modeling Framework und wurden im Rahmen des DFG Schwerpunktprojekts 1103 „Ein Prototyp für verteilte, interaktiv-kooperative Simulationen zur Beschleunigung von Entwurfszyklen im Konstruktiven Ingenieurbau“ und dem daraus folgenden Industrietransferprojekt entwickelt [15, 18, 38, 68]. So konnte ein virtueller Konstruktionsraum auf Basis von AutoCAD Architecture entwickelt werden, der um eine Vielzahl von Funktionalitäten erweitert wurde. Die CAD-gestützte Modellierung ermöglicht hierbei den Entwurf komplexer Geometrien und die Nutzung eines einheitlichen Gebäudemodells (BIM). Durch direkte Anbindung über ein Kommunikationsmodul können die Geometriedaten und Parameter direkt an die Simulation übertragen werden. Die zusätzlichen Parameter wie Randbedingungen und spezifische Materialkennwerte werden über das CAD-Werkzeug vorgegeben und können interaktiv verändert werden. Hierzu wurde AutoCAD u.a. um Module zum Pre-processing, zur Ereigniserkennung, für den Datentransfer sowie zur Aufbereitung des Datenmodells ergänzt (vgl. Abbildung 5.6). Diese Hauptkomponenten werden in den folgenden Abschnitten dargestellt.

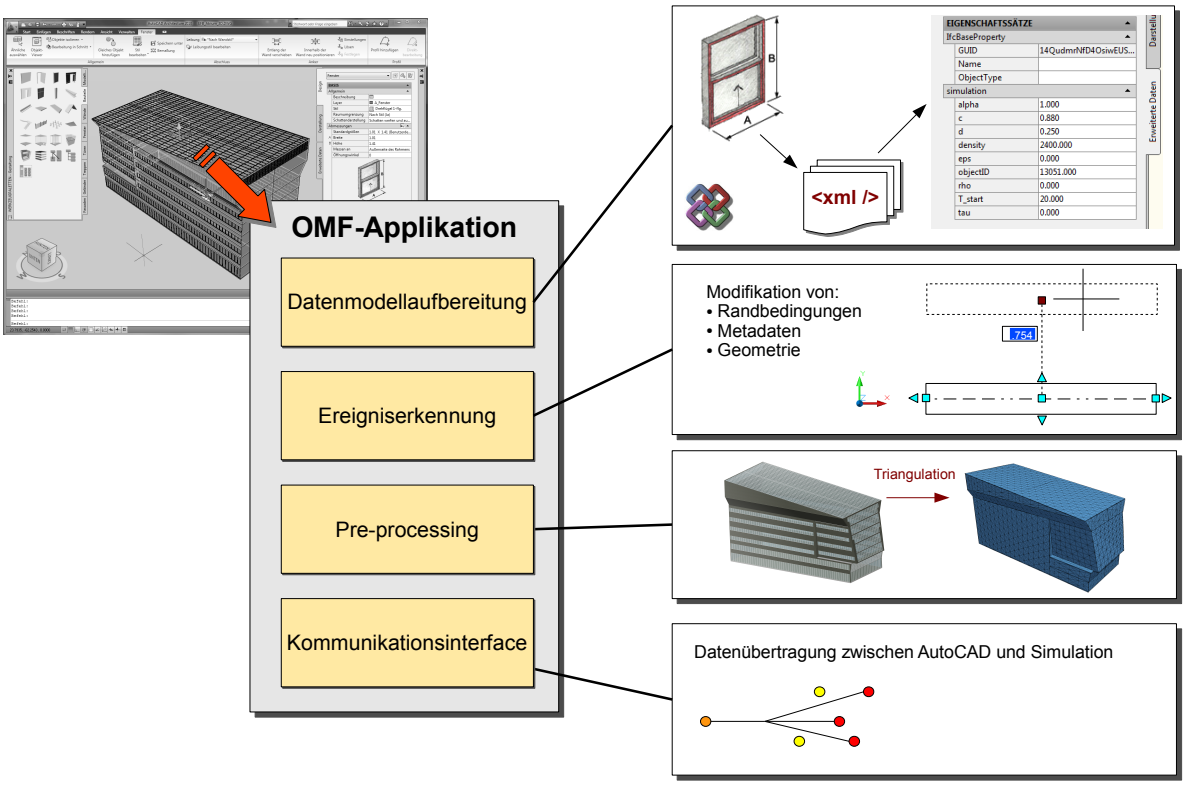


Abbildung 5.6: *AutoCAD Architecture Erweiterungen*

Datenmodellaufbereitung

Das entwickelte Modul zur Datenmodellaufbereitung erweitert die Bauteilobjekte in AutoCAD Architecture um für die Simulation notwendige Metainformationen, wie Randbedingungen und Materialeigenschaften (vgl. Tabelle 3.1). Diese benötigten Objektattribute werden über sogenannte Eigenschaftssätze (Property Sets) mit den Bauteilen verknüpft. Neue Attribute werden über die Eigenschaftssatz-Definition (in OMF durch die Klasse *AecDbPropertySetDef*) vorgegeben. Diese fassen die Merkmale einer Gruppe von Eigenschaften zusammen und sind konform zu den *IfcPropertySets*. Beispielsweise kann ein Eigenschaftssatz für ein Fenster erstellt werden, der Definitionen für Breite, Höhe und Unterkante der Brüstung enthält. Die Eigenschaftssätze können vom Benutzer über den AutoCAD Stil-Manager vorgegeben bzw. bearbeitet werden. Das entwickelte Modul unterstützt jedoch auch ein automatisiertes Anlegen und Verknüpfen von Eigenschaften mit den jeweiligen Bauteilobjekten. Hierzu können die Eigenschaftssätze über eine XML-Datei definiert werden. Beim Einfügen eines neuen Bauteils in die Zeichnung werden die Attribute (abhängig von der Definition in der XML-Datei) mit dem Objekt verknüpft. Jedes Attribut der Eigenschaftssatz-Definition setzt sich zusammen aus Namen, Beschreibung, Datentyp, Datenformat mit Grenzwerten und einem Vorgabewert. Alle Objekte werden permanent vom Programm überwacht und es wird kontrolliert, ob die benötigten Attribute vorliegen

und diese die vorgegebenen Grenzwerte nicht überschreiten. Die Bauteilattribute können vom Benutzer über das Eigenschaftsfenster in AutoCAD eingesehen und modifiziert werden (vgl. Abbildung 5.4).

Pre-processing

Das Pre-processing Modul übernimmt die Aufbereitung und Triangulation des AutoCAD Geometriemodells in ein Oberflächennetz. Für die geometrische Modellierung von Bauteilen und anderen 3D Objekten steht in AutoCAD Architecture der *AModeler* als Geometriekern zur Verfügung. Dieser ermöglicht die Generierung, Transformation und Verwaltung von Boundary Representation Modellen (b-rep). Ein Zugriff auf den *AModeler* ist über ObjectARX und OMF möglich. Geometrische Modelle werden im *AModeler* durch ihre begrenzenden topologischen verknüpften Oberflächen (Flächen, Kanten und Eckpunkte) beschrieben. Flächen sind hierbei immer eben, mit einem vom Körper nach außen zeigenden Normalenvektor. Gekrümmte Oberflächen werden im *AModeler* durch eine Facettierung angenähert. Mit steigender Anzahl der Facetten werden die Abweichungen zur ursprünglichen Geometrie geringer. Die Glättung gekrümmter Kanten kann in AutoCAD vom Benutzer durch die Variable *FACETDEV* gesteuert werden. Diese legt die Anzahl der auf gebogenen Objekten angezeigten Facetten fest.

Auf Basis des Geometriekerns konnte eine automatisierte Oberflächennetzgenerierung implementiert werden. Hierzu wird das Facettenmodell in ein Dreiecksnetz trianguliert, die begrenzenden Oberflächen also in einzelne Dreiecke zerlegt. Die Verknüpfung der Dreiecke mit ihren ursprünglichen Bauteilobjekten bleibt hierbei erhalten, so dass von jedem Dreieck auf das zugehörige Bauteil und die damit verbundenen Attribute (Materialeigenschaften und Randbedingungen) geschlossen werden kann. Dieses so generierte Dreiecksnetz stellt die Grundlage für den entwickelten thermischen Simulationskern dar (vgl. Kapitel 4 und Kapitel 3).

Kommunikationsinterface

Das Kommunikationsinterface ist die Schnittstelle zum Transfer des Simulationssetups, bestehend aus der Geometrie, den Materialeigenschaften und den Randbedingungen vom CAD-System an den Rechenkern. Es wurde auf Basis von Netzwerksockets realisiert. Ein Socket ist ein Software-Modul zum bidirektionalen Datenaustausch über ein Computernetzwerk. Für den entwickelten Ansatz wurde das Client-Server-Prinzip gewählt. Hierbei stellt der Server auf einem vorgegebenem Port einen Dienst (die thermische Simulation) zur Verfügung. Der Zugriff erfolgt durch den in AutoCAD implementierten Client über das lokale Netzwerk oder das Internet. Der gestartete Serverprozess wartet, bis er Daten von einem Client erhalten hat. Der Datentransfer wird vom Benutzer in AutoCAD ausgelöst, beispielsweise durch Interaktionen wie das Verschieben oder Hinzufügen von Bauteilen. Nach vollständiger Beendigung der Datenübertragung werden die Änderungen in der laufenden Simulation

berücksichtigt. Das Modul unterscheidet zwischen vier Kommunikationsprozessen, so dass die übertragenden Daten möglichst gering gehalten werden:

1. **Initialisierung**: Alle Geometrieobjekte, physikalischen Parameter und globalen Randbedingungen werden übertragen; das Oberflächennetz wird neu generiert; die Simulation wird gestartet.
2. **Änderung der Geometrie**: Nur die veränderten oder gelöschten Objekte werden übertragen; das Oberflächennetz wird neu generiert.
3. **Änderung von Objektattributen**: Ein geänderter physikalischer Parameter wird übertragen; das Oberfächennetz bleibt erhalten.
4. **Änderung von Randbedingungen**: Die geänderte Randbedingung wird übertragen; das Oberfächennetz bleibt erhalten.

Eine detaillierte Beschreibung des Übertragungsprotokolls kann [73] entnommen werden.

Ereigniserkennung

Zur Überwachung von Benutzerinteraktionen (wie das Verändern der Geometrie oder Objektattribute) in AutoCAD Architecture wurde ein Modul zur Ereigniserkennung entwickelt. Hierdurch wird, neben der Überwachung von Grenzwerten, eine automatisierte Datenübertragung, abhängig von Ereignissen (Events) innerhalb der CAD-Anwendung, möglich. Beispielsweise löst das Verschieben eines Bauteils eine Ereigniskette aus, bei der zuerst das Oberflächennetz generiert wird und mit den zugehörigen Attributen über das Kommunikationsinterface automatisch an die Simulation übertragen wird (vgl. vorherige Abschnitte).

Für die Überwachung von Systemereignissen und Benutzerinteraktionen stellen ObjectARX und OMF sogenannte *Reactor* Klassen zur Verfügung. Zur Implementierung eigener Überwachungswerkzeuge wird eine neue Klasse angelegt und von einem *Reactor* abgeleitet, durch Überschreiben der entsprechenden Event-Methoden kann dann mit eigenen Funktionalitäten auf die Ereignisse reagiert werden. Bei der Ereigniserkennung wird unterschieden zwischen dem *AcDbDatabaseReactor* zur Überwachung von Objektdatenbankereignissen (wie Löschen, Verändern, Hinzufügen von Objekten) und dem *AcEditorReactor* zum Detektieren von Systemereignissen (wie das Laden einer Zeichnung, etc.). Unter Verwendung dieser *Reactors* lässt sich eine komplexe Ereigniserkennung in AutoCAD Architecture umsetzen, die eine Überwachung von Grenzwerten und Objektpositionen im Entwurfsraum (Plausibilitätsprüfungen) sowie eine automatisierte Datenübertragung an den Simulationskern ermöglicht.

5.3 Verteilte Visualisierung auf Tiled-Display-Systemen

Durch die permanent steigende Rechenleistung von Computern und dem Einsatz von Hochleistungsrechnern werden die bei Computersimulationen anfallenden Datenmengen in der

Industrie und Wissenschaft immer größer. Für diese oft mehrere 100 Gigabyte großen Datensätze werden automatisierte effiziente Ansätze zur grafischen Darstellung benötigt, so dass die Daten ausgewertet und analysiert werden können. Die wissenschaftliche Visualisierung (scientific visualization) ist ein Teilgebiet der Informatik, welches sich mit Techniken zur Präsentation und Darstellung solcher Daten beschäftigt [78].

Im Vordergrund bei den in dieser Arbeit betrachteten thermischen Simulationen steht die Visualisierung von Skalarwerten, u.a. Temperaturfelder und mit Temperaturen versehene Oberflächen. Diese Werte haben immer einen eindeutigen geometrischen Bezug. Temperaturen lassen sich am besten farbig darstellen. Hierbei ist es sinnvoll, hohen Temperaturen die Farbe Rot zuzuordnen und für niedrige (kalte) Temperaturen die Farbe Blau zu verwenden. Ein verbreitetes Verfahren zur Darstellung von Datensätzen mit skalaren Attributen erfolgt über die Definition einer Farbtabelle [12]. In dieser wird eine vorgegebene Anzahl an Farben in Abhängigkeit von einem minimalen und einem maximalen Wert gespeichert. Die dazwischen liegenden Farben werden gleichmäßig abgestuft. Den Attributen des Datensatzes können dann die entsprechenden Farben aus der Farbtabelle zur Visualisierung zugeordnet werden.

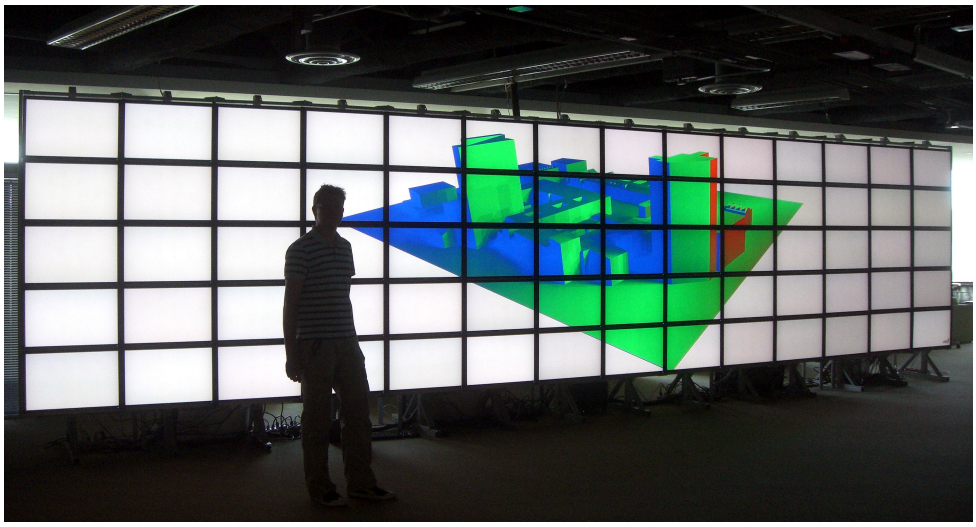


Abbildung 5.7: *Tiled-Display-Umgebung am Calit2 bestehend aus 70 gekoppelten 30 Zoll Displays*

Die entwickelte Simulationsumgebung stellt durch die Integration von Simulation und Visualisierung sehr hohe Anforderungen an die unterstützende Hardware und Algorithmik. Vor diesem Hintergrund wurde ein neuer Ansatz entwickelt, der neben der parallelen Simulation auch die Visualisierung verteilt auf einem Computercluster durchführt. Hierbei werden die Simulationsergebnisse direkt auf den Clusterknoten gerendert, so dass keine zusätzliche Datenkommunikation zwischen Simulation und Visualisierung notwendig ist. Durch diesen Ansatz ist es möglich, die in der thermischen Simulation in jedem Zeitschritt anfallenden großen Datenmengen in Echtzeit zu visualisieren und interaktiv in die Simulation einzugreifen. Die Visualisierung erfolgt hierbei in einer Tiled-Display-Umgebung (einer

Bildschirmwand bestehend aus vielen gekoppelten Displays vgl. Abbildung 5.7). In den folgenden Abschnitten wird der Ansatz zur verteilten Visualisierung thermischer interaktiver Simulationen auf Basis des CGLX-Frameworks [27] erläutert.

5.3.1 Die CGLX Architektur

Die Cross-Platform Cluster Graphic Library (CGLX) ist eine OpenGL basierte Grafikbibliothek für skalierbare verteilte Visualisierungssysteme (wie Multi-Projektor-Umgebungen und Tiled-Display-Systeme) und stellt ein intuitives Framework zur Entwicklung von Anwendungen für Multi-tile-Displayumgebungen zur Verfügung [36]. Einige Merkmale von CGLX sind eine hohe Skalierbarkeit und Performance auch auf heterogenen Systemen, ein hardware- und plattformunabhängiges Design sowie die Unterstützung für synchrone und asynchrone Benutzerinteraktionen. Mit CGLX lassen sich verteilte cluster-basierte Visualisierungssysteme betreiben und steuern. Des Weiteren stellt das Framework eine Programmierschnittstelle (API) zur Verfügung, welche die Entwicklung von parallelen hardware-beschleunigten OpenGL Anwendungen ermöglicht.

Die aktuelle CGLX Version 1.2.2 [27] unterstützt UNIX-Plattformen sowie Mac OS. CGLX dient als Middleware zur Kommunikation zwischen den einzelnen Clusterknoten und zur Synchronisation der Darstellung auf den Displays. Hierzu ist CGLX als dynamische Bibliothek (cglXlib) implementiert und wird von den CGLX Anwendungen und Tools benötigt (vgl. Abbildung 5.8). Diese verwaltet die laufende Anwendung auf dem Cluster, stellt die Schnittstelle für Anwendungsentwickler bereit und kontrolliert die Netzwerkressourcen und Benutzerinteraktionen.

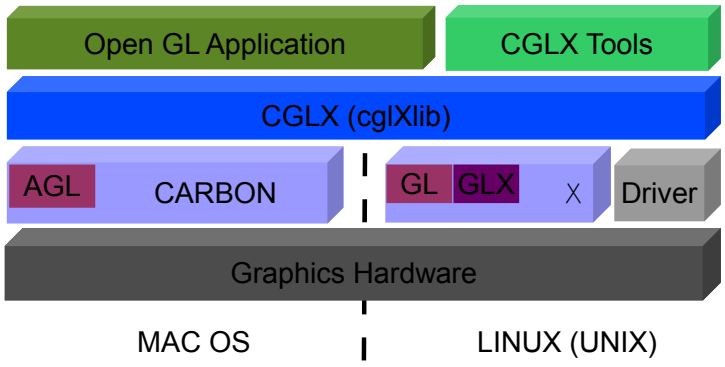


Abbildung 5.8: *Middleware Architektur aus [36]*

CGLX startet eine OpenGL Anwendungen auf jedem Knoten des Computer-Clusters. Zur Darstellung auf dem Tiled-Display-System wird die Szene abhängig von der Anordnung und Anzahl der einzelnen Bildschirme auf ein sogenanntes Visualisierungsgitter zerlegt (in Abbildung 5.7 beispielsweise in 70 rechteckige Zellen (Teilszenen)). Die auf den einzelnen Knoten laufenden OpenGL Anwendungen halten die gesamte Szene vor und nur durch geschickte Anpassung der jeweiligen Sicht auf die Teilszene erhält man eine konsistente Darstellung

auf dem gesamten Tiled-Display-System. CGLX berechnet hierbei die korrekten Projektions- und Transformationsmatrizen für jede Teilszene, um eine durchgehende Darstellung des Visualisierungsgitters zu ermöglichen. Intern verteilt und synchronisiert das Framework diese Darstellungsinformationen über die Knoten des Clusters. Hierzu laufen leichtgewichtige Kommunikationsthreads auf Basis von UDP Multicast Sockets auf den einzelnen Knoten des Visualisierungsgitters, die für den Datenaustausch zuständig sind. Durch diesen Ansatz brauchen auf jedem Knoten nur Teile der Szene gerendert werden. Außerdem ist ein effektives Culling (Überprüfung der Sichtbarkeit von Objekten) und eine effiziente Lastverteilung möglich. Dies erlaubt eine schnelle Ausgabe und Manipulation großer Datenmengen. CGLX unterstützt außerdem Benutzerinteraktionen wie Maus- oder Tastatureingaben, welche effizient auf die Tiled-Display-Umgebung verteilt werden.

Clusterbasierte Multi-Tile-Displays erlauben ein Echtzeitrendering großer Datenmengen mit hohen Frameraten und schnelle interaktive Manipulation der Daten. Somit ist das CGLX Framework ideal geeignet für die Entwicklung eines verteilten interaktiven Entwurfsraums zur kooperativen Planung von Gebäuden.

5.3.2 Anbindung des Simulationsframeworks an CGLX

Zur Kopplung des Simulationsframeworks an die verteilte Visualisierung auf Tiled-Display-Systemen auf Basis von CGLX wurde ein Ansatz basierend auf Netzwerk-Sockets entwickelt. Ein Netzwerk Socket ist ein Endpunkt einer bidirektionalen Interprozesskommunikation über ein Computernetzwerk. Alle Daten zwischen Simulation und Visualisierung werden über ein sogenanntes Loopback Interface ausgetauscht. Hierbei handelt es sich um eine virtuelle Netzwerk Schnittstelle, die vollständig in die Netzwerk Infrastruktur integriert ist und zur Kommunikation zwischen Prozessen auf demselben Computer verwendet werden kann. Somit laufen Simulation und Visualisierung in getrennten Prozessen und können unabhängig voneinander gestartet werden. Abbildung 5.9 zeigt die Kommunikationsschnittstellen des entwickelten Prototypen. Für den verteilten thermischen Simulationskern wird hier ein einfacher paralleler Radiosity Ansatz verwendet, bei dem Shooting und Formfaktorberechnung verteilt auf dem Computercluster ausgeführt werden. Zur Berechnung des Strahlungsaustausches werden hierbei die Oberflächen der Umgebung gleichmäßig auf die Clusterknoten verteilt und dort berechnet. Für die Simulation ist es notwendig, dass die Daten in jedem Zeitschritt auf allen Knoten vorliegen. Dies ist auch eine Voraussetzung für das verteilte Rendering mit CGLX. Die Datenkommunikation und Synchronisation der einzelnen Prozesse erfolgt hierzu mit einem effizienten Multicast-Verfahren auf Basis von UDP-Sockets. Weitere Ansätze für ein paralleles hierarchisches Radiosity wurden im Rahmen dieser Arbeit nicht weiter verfolgt, hierzu sei auf die Arbeiten von Bohn [21], Podehl [94] and Sinop [106] verwiesen.

Für die 3D Visualisierung der Simulationsergebnisse wurde ein Viewer basierend auf dem OpenSource Toolkit OpenSceneGraph [88], einem objekt-orientiertem OpenGL Framework

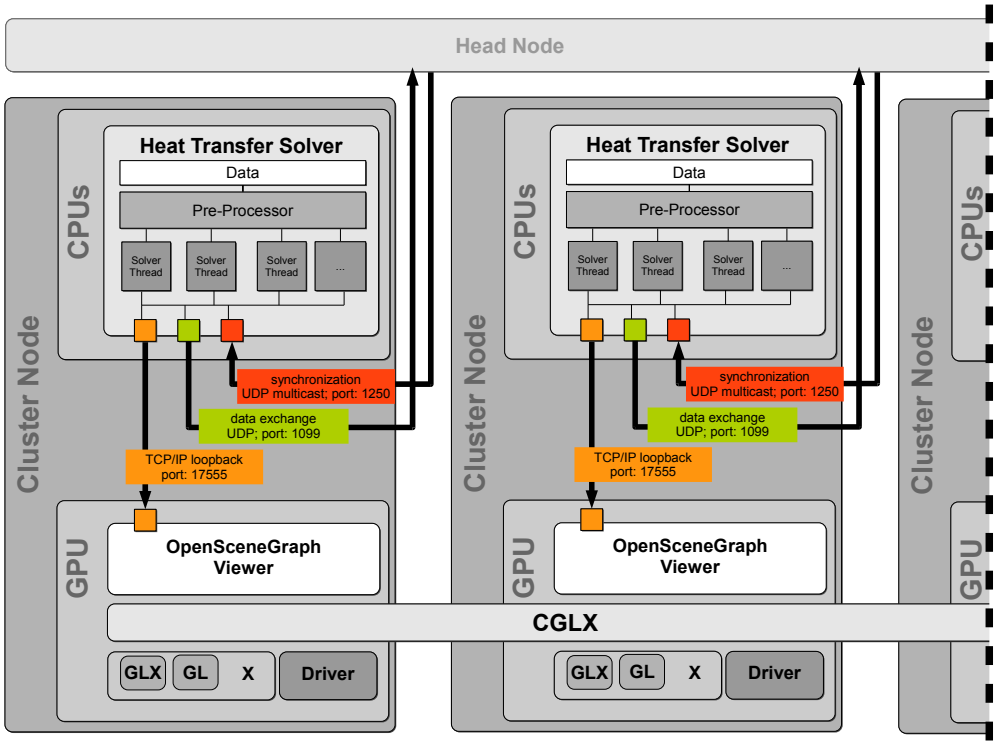


Abbildung 5.9: *Kommunikationsinterface der Simulationsumgebung*

entwickelt, das als CGLX-Erweiterung auf einem Visualisierungscluster gestartet werden kann. Der Viewer beinhaltet neben der Darstellung von Oberflächennetzen und damit verknüpften skalaren Attributen auch Standard Manipulationsmöglichkeiten, beispielsweise zum Ändern der Ansicht (u.a. Zoom, Rotation, etc.). Geometrieänderungen, und die in jedem Zeitschritt neu berechneten Substitutionsergebnisse (wie Temperaturen, Wärmeflüsse, etc.) werden automatisch über das Loopback Interface an den Viewer übertragen, der eine Aktualisierung der Darstellung auslöst. Hierzu ist im OpenSceneGraph Viewer ein Kommunikationsthread implementiert, der diese Daten entgegennimmt und verarbeitet.

5.3.3 Benchmark der verteilten Simulationsumgebung

In diesem Abschnitt wird ein Benchmark der Simulationsumgebung vorgestellt. Als Testumgebung dient die HIPerSpace (Highly Interactive Parallelized Display Space) am Calit2 [26] bestehend aus 70 Bildschirmen mit 286 Megapixel Auflösung, gesteuert durch einen Computercluster mit 18 Dell XPS 710 Knoten (Intel Core2 Q6600 Quad-Cores und 4GB DDR3 Speicher) verbunden über ein Gigabit Netzwerk. Der Benchmark zeigt die parallele Effizienz und Frameraten des entwickelten Prototypen. Als Szenario ist die Simulation der Energieverteilung durch direkte Sonneneinstrahlung auf ein Stadtmodell gewählt. Für die Diskretisierung des Oberflächennetzes werden Auflösungen mit 10000, 120000 und 480000 Dreiecken be-

trachtet. Abbildung 5.10 zeigt die Simulation auf der HIPerSpace, farbig dargestellt ist die Energieverteilung auf den Oberflächen.

Die parallele Effizienz (für eine steigende Anzahl an verwendeten Knoten) ist in Abbildung 5.11 gegeben. Unter Auslastung des gesamten Clusters (72 CPUs) konnten konstante Frameraten von ca. 50 fps bei ca. 15 Updates der Simulation pro Sekunde auf dem Tiled-Display-System erreicht werden. Hiermit konnte gezeigt werden, dass der Prototyp für komplexe Geometrien und hohen Aktualisierungsraten der Simulation in der Lage ist, die anfallenden Daten parallel und mit hohen Frameraten zu rendern. Somit stellt die Simulationsumgebung eine geeignete Plattform für kooperativ interaktive Planungsprozesse an komplexen Gebäuden dar.

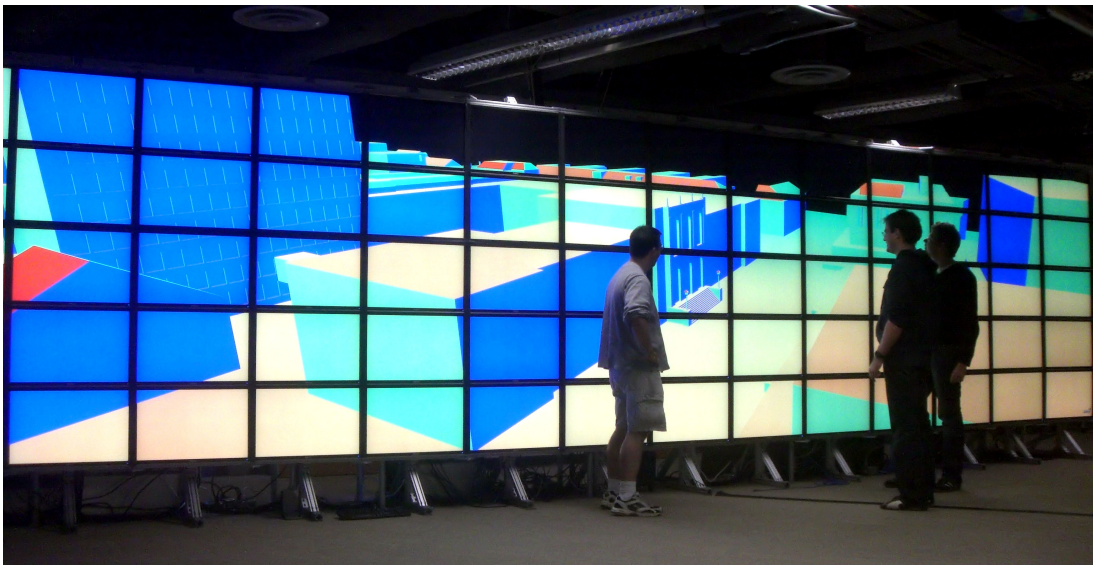


Abbildung 5.10: *Thermische Simulation auf HIPerSpace am Calit2*

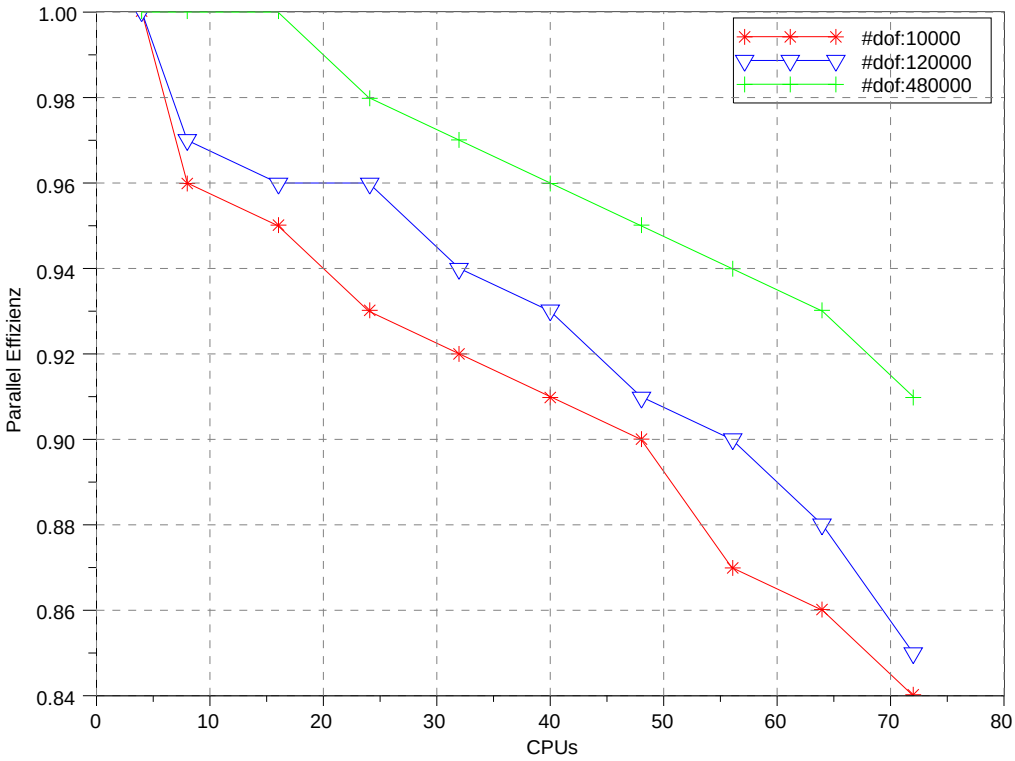


Abbildung 5.11: *Parallele Effizienz auf 18 Clusterknoten mit je 4 CPU´s*

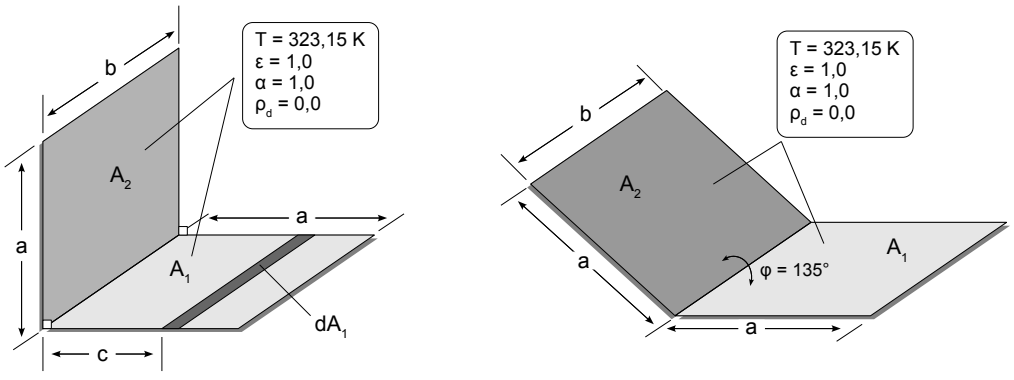
6 Validierung und Anwendungen

In diesem Kapitel werden verschiedene Benchmarks zur Validierung des entwickelten Simulationskerns sowie einige komplexe Anwendungsbeispiele aus dem Bau- und Umweltingenieurwesen betrachtet. Diese reichen von Behaglichkeitsoptimierungen in Innenräumen über thermische Optimierungen von Gebäudestrukturen bis hin zur effizienten Planung von Solaranlagen. Hierzu wurde der Prototyp erfolgreich in diversen Industrieprojekten projektbegleitend als Planungswerkzeug eingesetzt.

6.1 Validierung

6.1.1 Strahlung zwischen rechteckigen Platten

Für das erste Validierungssetup werden zwei orthogonal zueinander angeordnete rechteckige Platten der selben Länge, mit einer gemeinsamen Kante betrachtet [19]. Eine detaillierte Beschreibung des Setups ist in Abbildung 6.1a gegeben. Die beiden Platten sind in einem die Strahlung nicht beeinflussenden Medium platziert und haben dieselben Materialeigenschaften. Das Ergebnis der Simulation ist die Verteilung der Strahlungsstromdichte über die Platte durch den von der anderen Platte ausgehenden Strahlungsfluss. In diesem Setup werden unterschiedliche Diskretisierungen (Auflösungen) der Platten (mit: 2, 8, 32, 128, 512 und 2048 Dreiecken) verwendet. Die Simulationsergebnisse werden mit einer Refe-



(a) *Zwei orthogonale rechteckige Platten* ***(b)*** *Zwei Platten mit* $\varphi = 135°$

Abbildung 6.1: *Setup der Validierung*

renzlösung verglichen, die man für dieses geometrisch einfache Setup aus der analytischen

Lösung des Formfaktorintegrals (vgl. Gleichung 2.16) erhält [57]. Sie ist für einen schmalen Streifen und ein Rechteck in Tabelle 2.1 gegeben. Um hieraus die Verteilung über die Platte zu erhalten, kann F_{ij} für mehrere differentielle Streifen ausgewertet werden. Das Ergebnis der Konvergenzstudie ist in Abbildung 6.3 dargestellt. Hierbei ist die relative Fehlernorm als Summe des relativen Fehlers geteilt durch die Referenzlösung definiert:

$$\varepsilon = \frac{1}{n} \sum_{x}^{n} \frac{\left| B_{ref}(x) - B(x) \right|}{B_{ref}(x)}, \tag{6.1}$$

mit der Anzahl der Dreicke n. Wie Abbildung 6.3 zu entnehmen ist, wird eine Konvergenzrate zweiter Ordnung im Raum erreicht. Abbildung 6.2 zeigt das Ergebnis der Simulation. Abgebildet ist neben der farbigen Energieverteilung auch die Struktur des Oberflächennetzes, welches eine adaptive Verfeinerung an Stellen mit hohen Energiegradienten zeigt.

Im zweiten Benchmark werden zwei rechteckige Platten der selben Länge, angeordnet in einem Winkel von $\varphi = 135°$, betrachtet. Die Konfiguration des Setups ist in Abbildung 6.1b dargestellt. Die beiden Platten sind in einem die Strahlung nicht beeinflussendem Medium platziert und haben dieselben Materialeigenschaften. Die Geometrie ist hier mit 256 Dreiecken aufgelöst. Die Verteilung der Strahlungsstromdichte über die Platte ist zusammen mit der Referenzlösung in Abbildung 6.4 dargestellt. Die Referenzlösung für dieses Setup konnte durch eine numerische Integration über das Formfaktorintegral mit einem Computeralgebrasystem gefunden werden [76]. Das Ergebnis zeigt nur geringe Abweichungen ($< 1,0\%$) zwischen Referenzlösung und Simulationsergebnis.

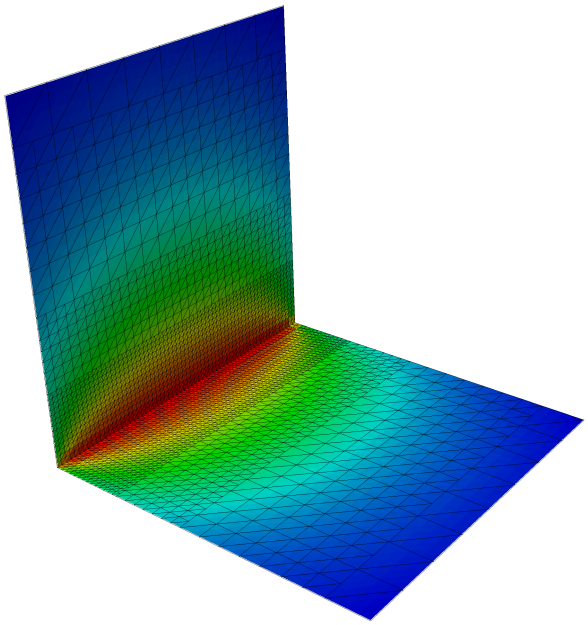


Abbildung 6.2: *Energieverteilung über zwei rechteckige orthogonale Platten*

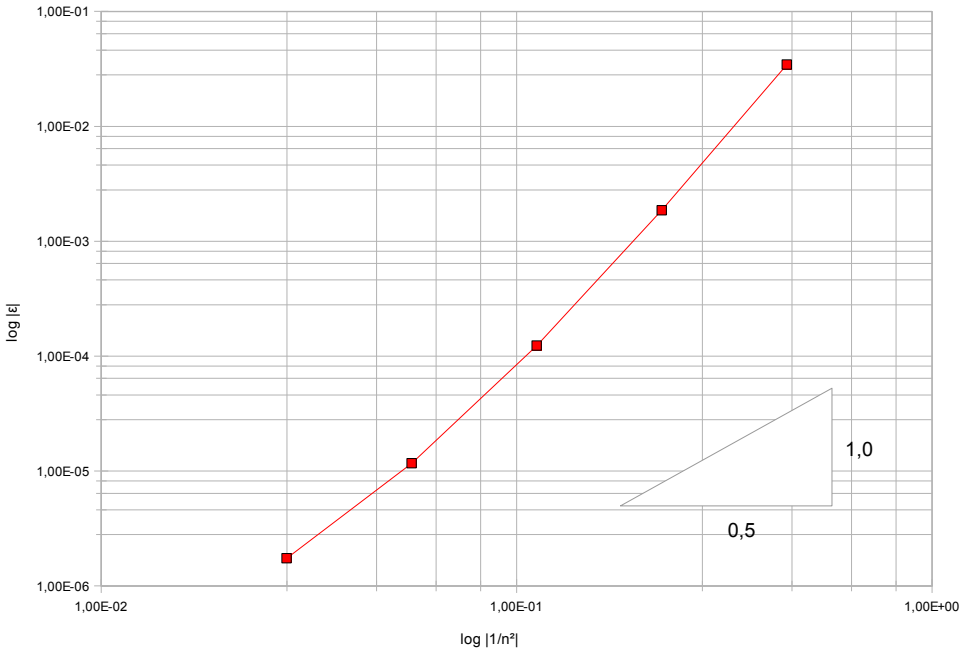


Abbildung 6.3: *Räumliche Konvergenz (zwei rechteckige orthogonale Platten)*

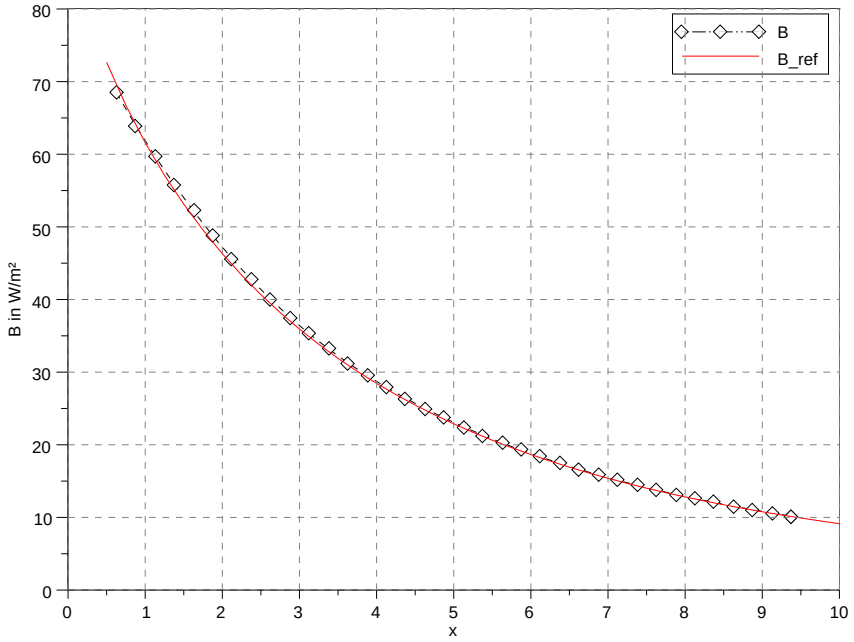


Abbildung 6.4: *Simulationsergebnisse und Referenzlösung (zwei rechteckige Platten mit Winkel φ)*

6.1.2 Strahlung zwischen Zylinder und Platte

Dieser Benchmark untersucht die Genauigkeit des Verfahrens im Zusammenhang mit gekrümmten Oberflächen. Hierzu wird die Strahlungsstromdichte ausgehend von der Oberfläche eines Zylinders auf eine rechteckige Platte untersucht [19]. Zylinder und Platte haben die selbe Höhe h und überschneiden sich nicht. Eine detaillierte Beschreibung kann Abbildung 6.5 entnommen werden. Für den Zylinderradius sind $r = 4,0$, für Abstand s, Höhe h und Breite b 10,0 gewählt. Die beiden Objekte sind in einem die Strahlung nicht beeinflussenden Medium platziert und haben die selben Materialeigenschaften. Die Berechnung wird für unterschiedliche Diskretisierungen der Oberflächen durchgeführt.

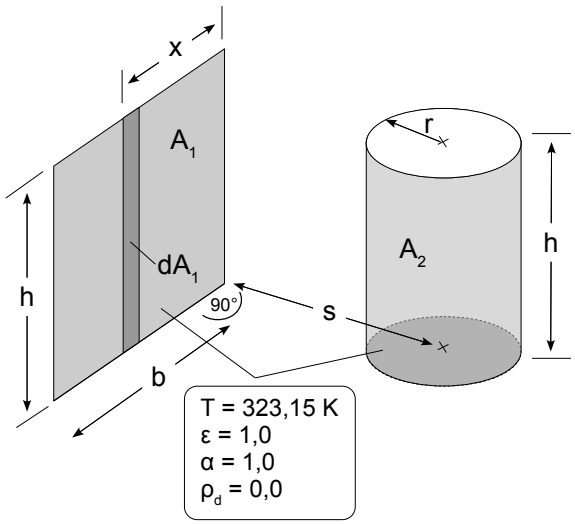


Abbildung 6.5: *Setup: Strahlung zwischen Zylinder und Platte*

Für dieses Setup ist eine analytische Lösung nach Howell [57] gegeben:

$$F_{d12} = \frac{S}{S^2+X^2}\left(1-\frac{1}{\pi}\left[cos^{-1}\frac{B}{A}-\frac{\sqrt{A^2+4H^2}}{2H}cos^{-1}\frac{B}{AC}-\frac{B}{2H}sin^{-1}\frac{1}{C}\right]-\frac{A}{4H}\right), \tag{6.2}$$

mit:

$$\begin{aligned} S = s/r,\ \ X = x/r,\ \ H = h/r \\ A = H^2+S^2+X^2-1 \\ B = H^2-S^2-X^2+1 \\ C = \sqrt{S^2+X^2}. \end{aligned} \tag{6.3}$$

Das Ergebnis der Konvergenzstudie ist in Abbildung 6.6 dargestellt. Die relative Fehlernorm ist wie im vorherigen Abschnitt definiert (vgl. Gleichung 6.1). Für die Referenzlösung wird Gleichung 6.2 für mehrere differentielle Streifen auf der Platte ausgewertet. Auch bei diesem Beispiel konnte eine Konvergenzrate zweiter Ordnung im Raum erreicht werden. In Abbildung 6.7 ist die Energieverteilung aus der Simulation über die Platte dargestellt.

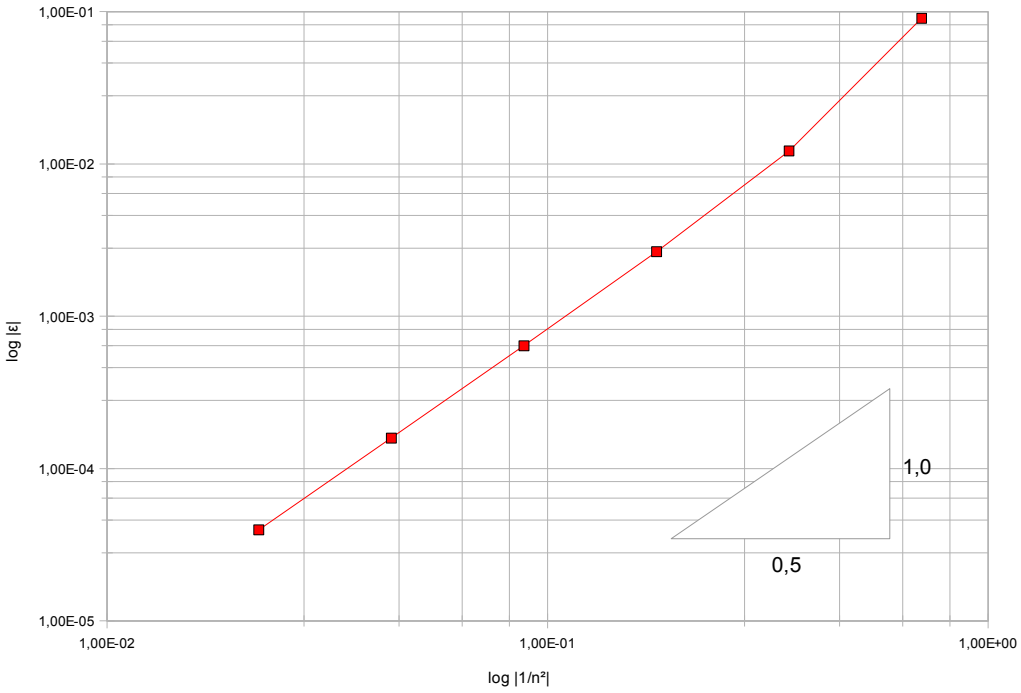


Abbildung 6.6: *Räumliche Konvergenz (Strahlung zwischen Zylinder und Platte)*

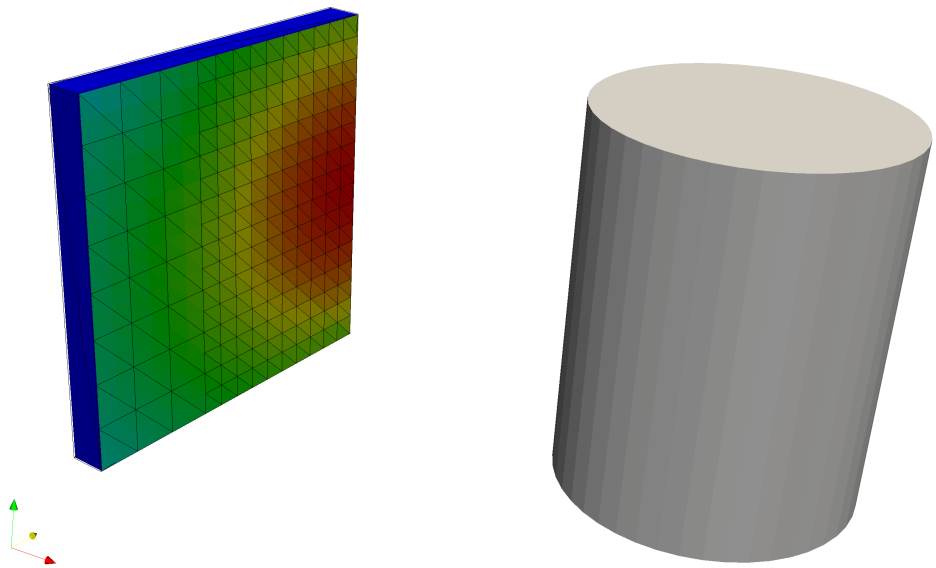


Abbildung 6.7: *Energieverteilung aus Simulation über die Platte*

6.1.3 Strahlung zwischen Kugel und differentiellem ebenem Element

Auch in diesem Benchmark wird die Genauigkeit des Verfahrens im Zusammenhang mit gekrümmten Oberflächen betrachtet. Hierzu wird die Strahlungsstromdichte ausgehend von der Oberfläche einer Kugel auf ein differentielles ebenes Element untersucht. Die Normale des Elements verläuft durch den Mittelpunkt der Kugel mit einem Radius von $r = 5,0$. Der Abstand zwischen Kugelmittelpunkt und Element beträgt $h = 10,0$. Die beiden Objekte sind in einem die Strahlung nicht beeinflussenden Medium platziert und haben die selben Materialeigenschaften. Das Benchmarksetup ist in Abbildung 6.13 dargestellt. Die Simulation wird für unterschiedliche Diskretisierungen der Oberflächen durchgeführt und mit der analytischen Lösung nach Howell [57] verglichen:

$$F_{d12} = \frac{r^2}{h^2}. \tag{6.4}$$

Das Ergebnis mit einer Konvergenzrate zweiter Ordnung im Raum ist in Abbildung 6.9 aufgeführt.

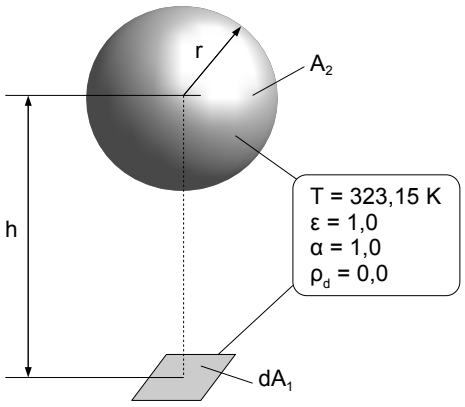


Abbildung 6.8: *Setup: Strahlung zwischen Kugel und differentiellem ebenem Element*

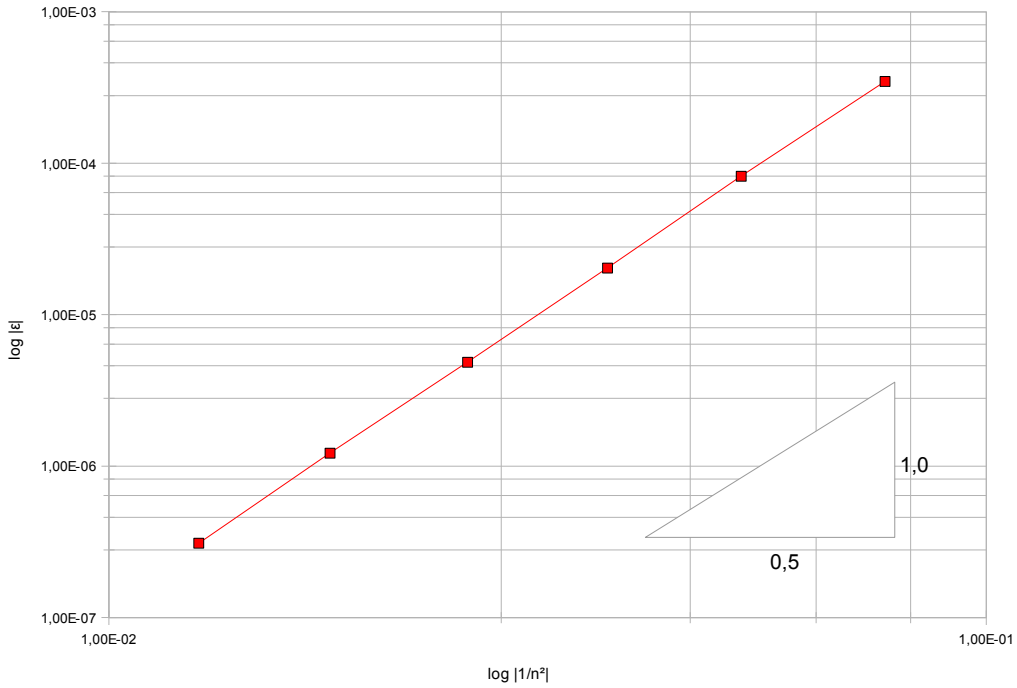


Abbildung 6.9: *Räumliche Konvergenz (Strahlung zwischen Kugel und differentiellem Element)*

6.1.4 Wärmeleitung in mehrschichtigen Bauteilen

In diesem Validierungsbeispiel wird die Gültigkeit des entwickelten Finite-Differenzen-Verfahrens zur Berechnung der Wärmeleitung in gekoppelten Bauteilen mit unterschiedlichen Gitterauflösungen und Leitfähigkeiten bewiesen. Betrachtet wird der Temperaturverlauf in einem mehrschichtigen Wandaufbau bestehend aus vier Schichten (vgl. Abbildung 6.10). Diese setzen sich aus einem Innenputz mit $\lambda_1 = 0,40 W/(mK)$, einem Porenbetonmauerwerk mit $\lambda_2 = 0,24 W/(mK)$, einer Wärmedämmschicht mit $\lambda_3 = 0,04 W/(mK)$ sowie einem Außenputz mit $\lambda_4 = 0,70 W/(mK)$ zusammen. Die Bauteile sind mit $10x20x40$, $10x20x40$, $20x10x40$ und $10x10x40$ Knoten diskretisiert. Als Randbedingungen werden an der Innen- und Außenwand Dirichlet-Randbedingungen mit konstanten Temperaturen von $T_i = 20,0°C$ und $T_e = -10,0°C$ angesetzt. Für die übrigen Ränder sind periodische Randbedingungen angesetzt, so dass sich ein quasi-eindimensionaler stationärer Temperaturverlauf einstellt der sich mit einer analytischen Lösung vergleichen lässt. Die analytischen Temperaturwerte an den Schichtgrenzen ergeben sich aus den Wärmedurchlasswiderständen R sowie der Wärmestromdichte q wie folgt [39]:

$$\begin{aligned} T_1 &= T_i - R_1 \cdot q = 19,360°C \quad \text{mit} \quad R_1 = d_1/\lambda_1 = 0,0625 K/W \\ T_2 &= T_1 - R_2 \cdot q = 10,832°C \quad \text{mit} \quad R_2 = d_2/\lambda_2 = 0,8333 K/W \\ T_3 &= T_2 - R_3 \cdot q = -9,635°C \quad \text{mit} \quad R_3 = d_3/\lambda_3 = 2,000 K/W \end{aligned} \tag{6.5}$$

mit

$$q = (T_i - T_e)/R_T = 10,234 W/m^2 \quad \text{mit} \quad R_T = R_1 + R_2 + R_3 + R_4. \tag{6.6}$$

Der Temperaturverlauf aus der Simulation und die Referenzlösung über die einzelnen Schichten sind in Abbildung 6.11 aufgeführt. Hierbei kann gezeigt werden, dass die Berechnung des linearen Temperaturverlaufs in mehrschichtigen Bauteilen mit unterschiedlichen Leitfähigkeiten, mit dem in Abschnitt 4.2.3 vorgestellten FDM-Ansatz, exakte Ergebnisse liefert. Abbildung 6.12 zeigt einen Ausschnitt aus dem dreidimensionalen Wandmodell mit den farbig aufgetragenen Temperaturen an den Gitterknoten.

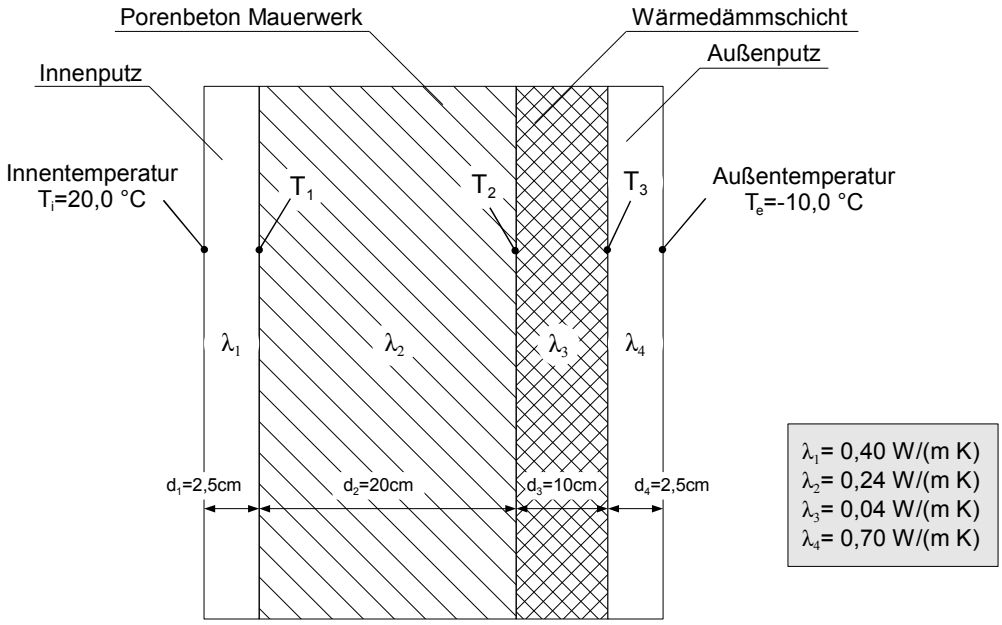


Abbildung 6.10: *Mehrschichtiger Wandaufbau*

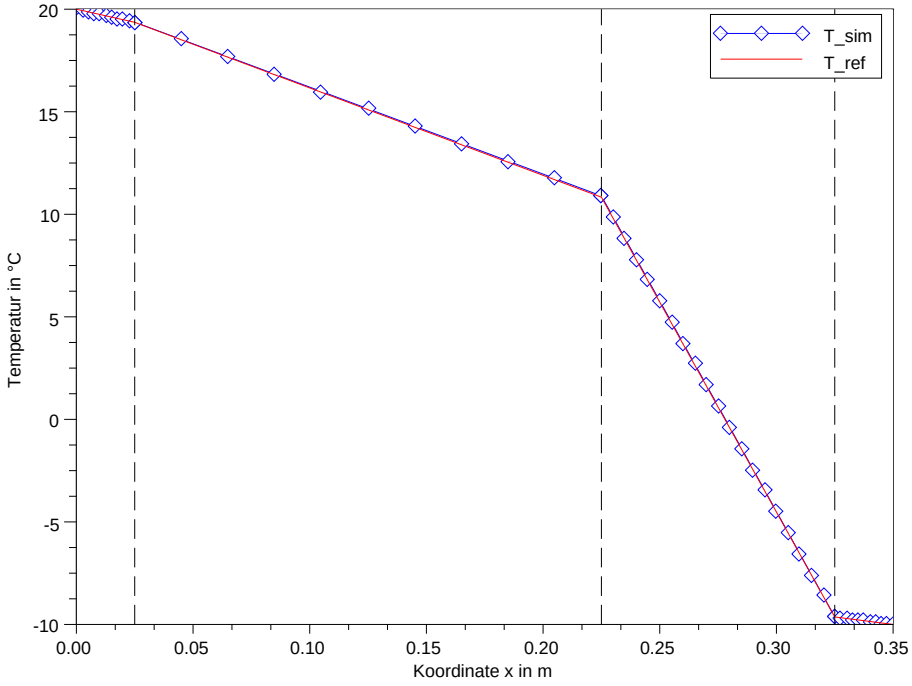


Abbildung 6.11: *Temperaturverlauf in mehrschichtiger Wand aus Simulation und analytischer Lösung*

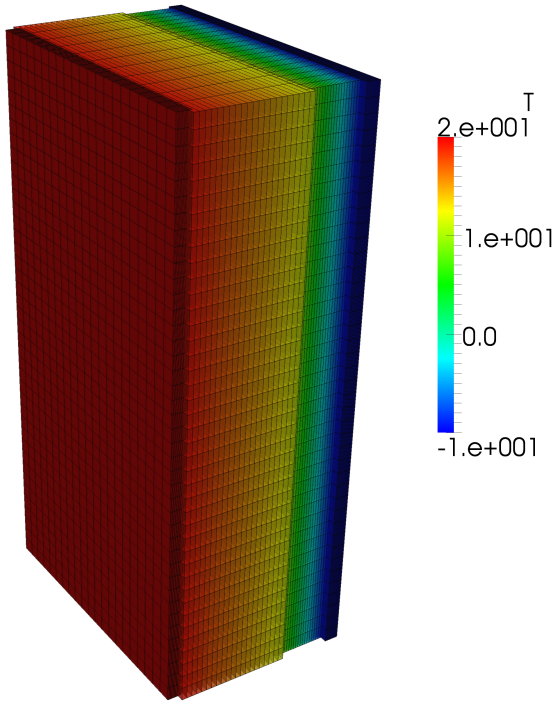


Abbildung 6.12: *Temperaturverlauf in mehrschichtiger Wand*

6.1.5 Kopplung Strahlung und Wärmeleitung

Zur Validierung der Kopplung von Strahlung und Wärmeleitung innerhalb einer Struktur wird eine isotrope Betonplatte, erwärmt durch eine konstante Strahlungsquelle, betrachtet. Der Aufbau der Anordnung ist im Detail in Abbildung 6.13 beschrieben. Die Betonplatte ist in einer abgeschlossenen Umgebung platziert, mit einer Strahlungsquelle auf der rechten Seite. Die linke Seite der Platte hat eine Dirichlet-Randbedingung, die rechte eine Neumann-Randbedingung, die übrigen Ränder sind periodisch.

Die Simulationsergebnisse wurden mit einer stationären analytischen Lösung verglichen. Hierzu vereinfacht sich für den eindimensionalen Fall das Fouriersche Gesetz der Wärmeleitung (vgl. Gleichung 2.23) zu:

$$q_x = \frac{\lambda}{L}(T_1 - T_0) \tag{6.7}$$

Aus Gleichung 6.7 und dem vom Körper ausgehenden Strahlungsfluss (gegeben durch das Stefan-Boltzmann-Gesetz, vgl. Gleichung 2.10) erhält man für das Temperaturgleichgewicht am rechten Rand:

$$q_{env} + q_{solar} - \varepsilon\sigma ATeq^4 = \frac{\lambda}{L}(T_1 - T_0) \tag{6.8}$$

Durch Lösen dieser Gleichung erhält man die Referenzlösung $T_{eq} = 72,4235°C$. Die Temperatur am rechten Rand aus der Simulation konvergiert gegen die analytische Lösung unabhängig von der Gitterauflösung. Somit kann gezeigt werden, dass die Kopplung von Strahlung und Wärmeleitung über Neumann-Randbedingungen für den linearen Temperaturverlauf exakte Ergebnisse liefert.

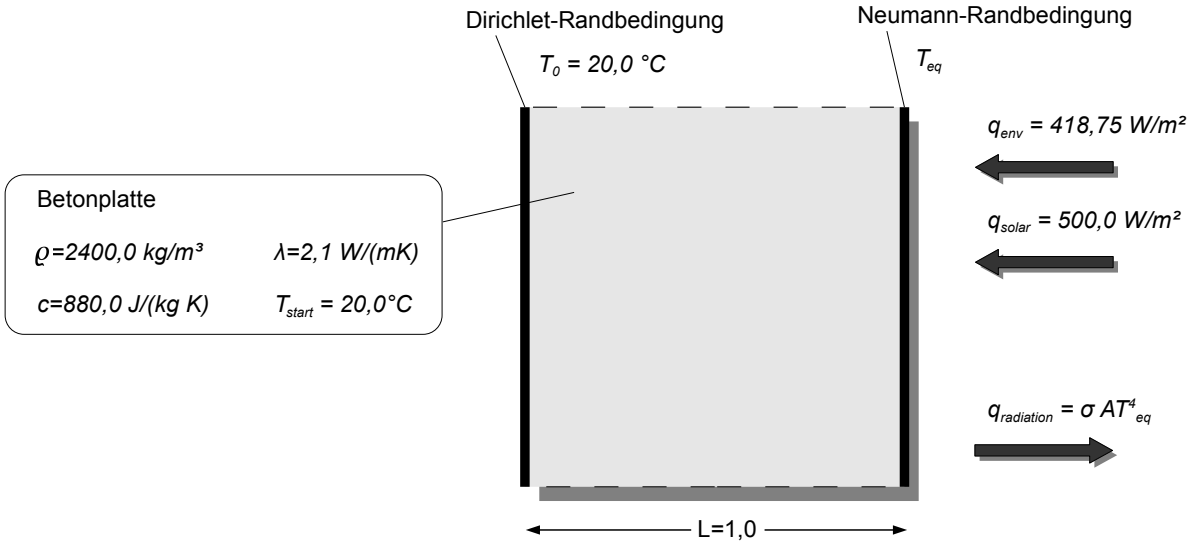


Abbildung 6.13: *Setup des Benchmarks: Kopplung Strahlung und Wärmeleitung*

6.2 Anwendungsbeispiele

6.2.1 Thermische Komfort-Simulation in einem Großraumbüro

Das in diesem Abschnitt vorgestellte Anwendungsbeispiel zeigt die Strahlungsenergieverteilung zur Optimierung des thermischen Komforts in einem Großraumbüro (vgl. Abbildung 6.14). Die thermische Behaglichkeit an Arbeitsplätzen ist die Grundlage für effizientes Arbeiten und muss bereits in der frühen Planungsphase eines Gebäudes berücksichtigt werden. Außerdem können durch passive und aktive Maßnahmen deutliche Reduktionen des Energieverbrauchs (von Heizungs- und Klimaanlagen) erreicht werden. Mit einer thermischen Simulation (wie sie hier durchgeführt wird) lässt sich der thermische Komfort für eine Gebäudestruktur optimieren. Hierbei können unterschiedliche bauliche Ausführungen untersucht und ihre Effizienz überprüft werden.

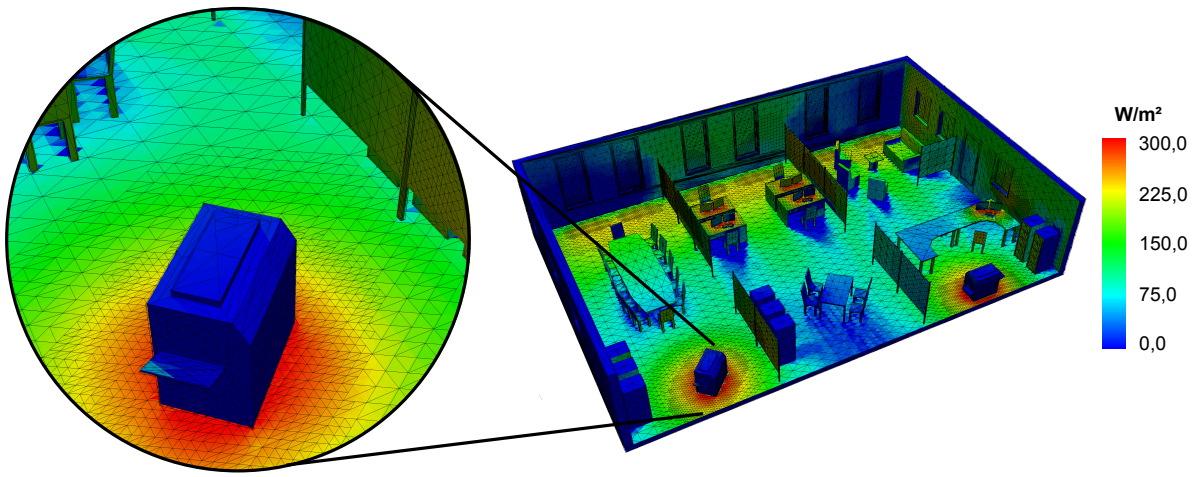


Abbildung 6.14: *Strahlungsenergieverteilung in einem Großraumbüro mit 250.000 Dreiecken diskretisiert*

In diesem Anwendungsbeispiel wird der Strahlungsaustausch in einem Großraumbüro mit folgenden Strahlungsquellen berechnet:

- Direkte Solarstrahlung mit $E = 700,0\ W/m^2$ durch die hinteren Fenster (mit einem Transmissionsgrad von $\tau = 0,8$ und einem Reflexionsgrad von $\rho = 0,2$).
- Mehrere Computerbildschirme mit $E = 400,0\ W/m^2$.
- Zwei Fotokopierer mit $E = 400,0\ W/m^2$.

Die Geometrie ist durch ein Oberflächennetz, bestehend aus ca. 10.000 Dreiecken, beschrieben und wird zur Laufzeit (bei der hierarchischen Radiosity Simulation) adaptiv auf ca. 250.000 Dreiecke verfeinert.

Ein Ergebnis der Simulation ist in Abbildung 6.14 gegeben. Dargestellt ist neben der farbigen Energieverteilung auch die Struktur des Oberflächennetzes. Gut zu erkennen ist die adaptive

Verfeinerung des Netzes an Stellen, an denen die Energiegradienten hoch sind. Die Laufzeit für diese Simulation beträgt ca. 150 Sekunden bei ca. 55MB Speicherbedarf auf einem Intel Core2 Q9550 Quad-Core.

6.2.2 Sonnenstrahlung auf ein Stadtmodell

Dieses Simulationsbeispiel zeigt die Energieverteilung durch direkte und diffuse Sonneneinstrahlung auf ein Stadtmodell mit ca. 1,0 Millionen Oberflächendreiecken. Die genauen Kenntnisse über die Energiedichten können bei der Planung und Optimierung von Solaranlagen sowie als Eingangsparameter für weitere bauphysikalische Fragestellungen dienen. Die Eingangsgeometrie wird zur Laufzeit adaptiv auf ca. 10 Millionen Dreiecke verfeinert. Als Randbedingungen wurden $450 W/m^2$ direkte Solarstrahlung und $250 W/m^2$ diffuse Solarstrahlung vorgegeben. Die Anfangstemperatur auf den Oberflächen beträgt $25,0°C$. In Abbildung 6.15 ist die Energieverteilung auf den Oberflächen für einen Sonnenstand farbig dargestellt. Die Durchführung der Simulation auf einem Intel Core2 Q9550 Quad-Core beträgt ca. 700 Sekunden bei einem Speicherbedarf von ca. 1.400MB.

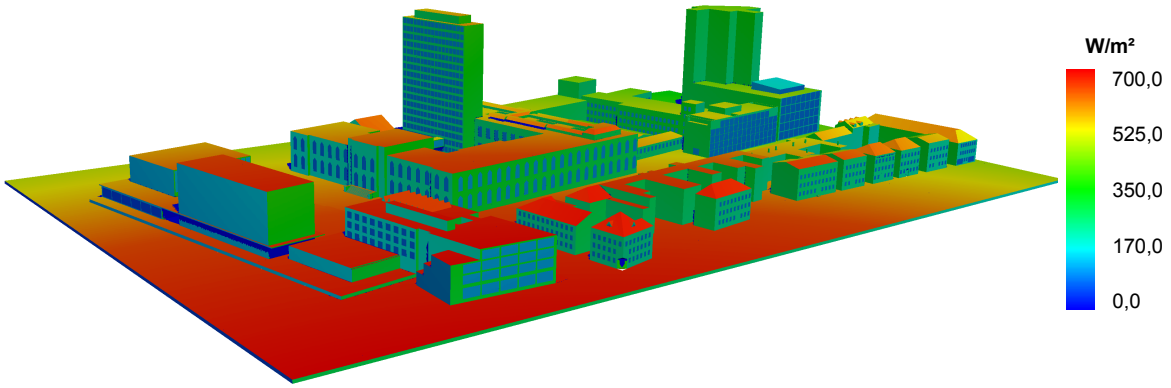


Abbildung 6.15: *Energieverteilung auf den Oberflächen induziert durch solare Strahlung*

6.2.3 Sonneneinstrahlung auf einen offenporigen Asphalt

In diesem Beispiel wird die Anwendbarkeit des gekoppelten Problems von Strahlung und Wärmeleitung an einer komplexen Geometrie demonstriert. Hierzu wird die Temperaturverteilung in einem offenporigen Asphalt (OPA), induziert durch Solarstrahlung für unterschiedliche Sonnenstände, untersucht. In Kombination mit einer Strömungssimulation lassen sich somit detaillierte Kenntnisse über die thermischen Verhältnisse innerhalb des porösen Mediums erlangen. Diese Simulationen unterstützen die Optimierung der Zusammensetzung und Eigenschaften von Asphalt. Das geometrische Modell wird durch Einscannen einer Asphaltprobe mit einen Computertomographen gewonnen und führt zu Netzen mit hohen Auflösungen. In diesem Fall wird ein Scan mit 400x400x350 Voxeln

erzeugt, aus dem ein Oberflächennetz mit ca. 1,0 Millionen Dreiecken (unter Verwendung des Marching-Cubes-Algorithmus) durch Triangulation gewonnen wird [1]. Diese hohe Auflösung ist notwendig, um die fein-granularen Porenstrukturen des porösen Asphalts aufzulösen. Das Oberflächennetz wird während der Simulation adaptiv auf 10 Millionen Dreiecke verfeinert. Das FDM-Rechengitter innerhalb der Struktur hat eine Auflösung von 200x100x100 Gitterknoten. Abbildung 6.16 zeigt die Temperaturverteilung über die Oberfläche sowie für einen Schnitt durch die Struktur für unterschiedliche Sonnenstände. Die Laufzeit für diese Simulation beträgt ca. 1.900 Sekunden bei ca. 1.750MB Speicherbedarf auf einem Intel Core2 Q9550 Quad-Core.

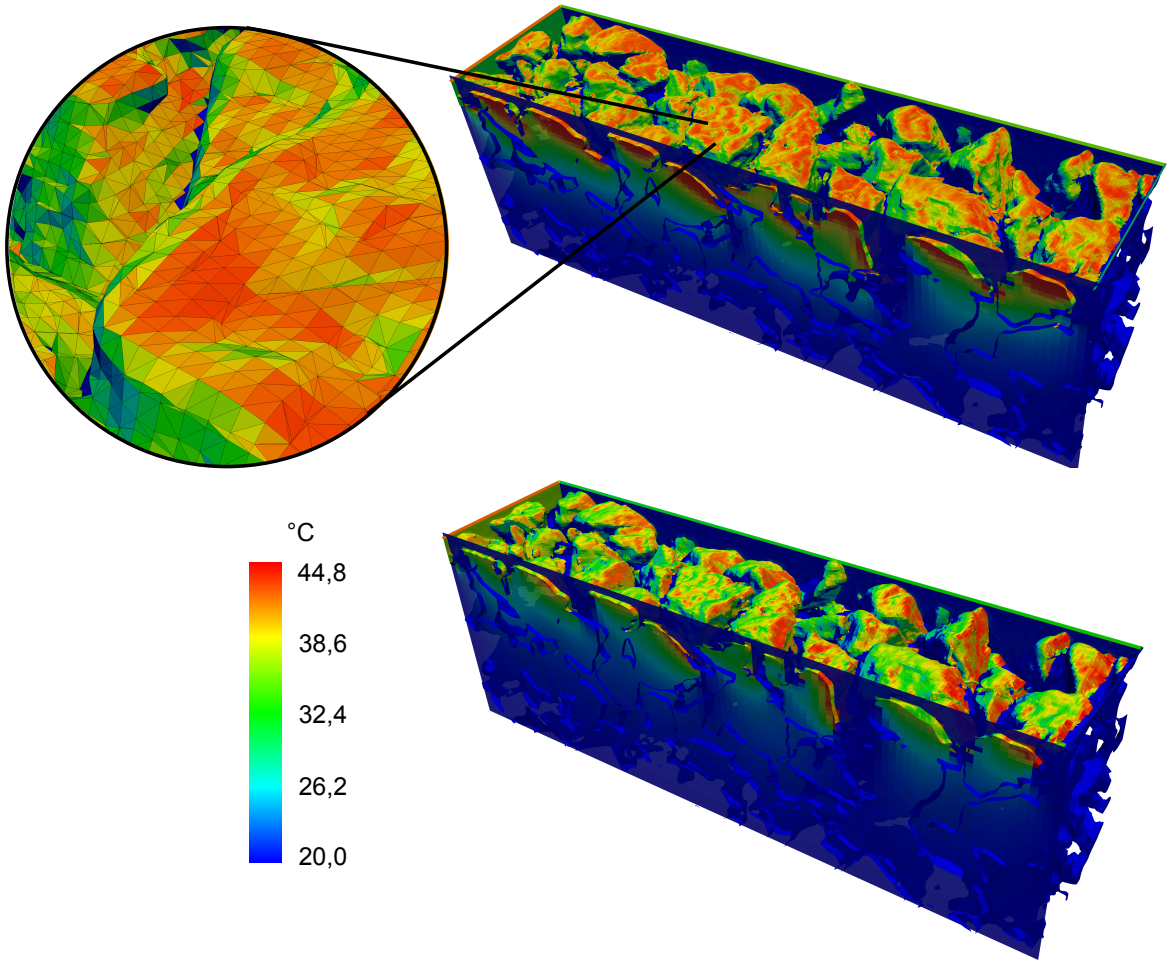


Abbildung 6.16: *Sonneneinstrahlung auf einen offenporigen Asphalt*

6.2.4 Temperaturverteilung an einem Wohnhaus

Mit diesem Anwendungsbeispiel wird die Temperaturentwicklung in und an einem Wohnhaus über einen Zeitraum von 48 Stunden simuliert. Das Gebäude besteht aus einem Holzsatteldach mit einer Dachneigung von 8°, einem zweischichtigem Wandaufbau, der sich

aus einer Wärmedämmschicht und einem Lochziegelmauerwerk zusammensetzt sowie vier Fenstern mit einer Sturzhöhe von 2,0m. Der Gebäudeaufbau sowie alle benötigten Materialeigenschaften sind in Abbildung 6.17 gegeben.

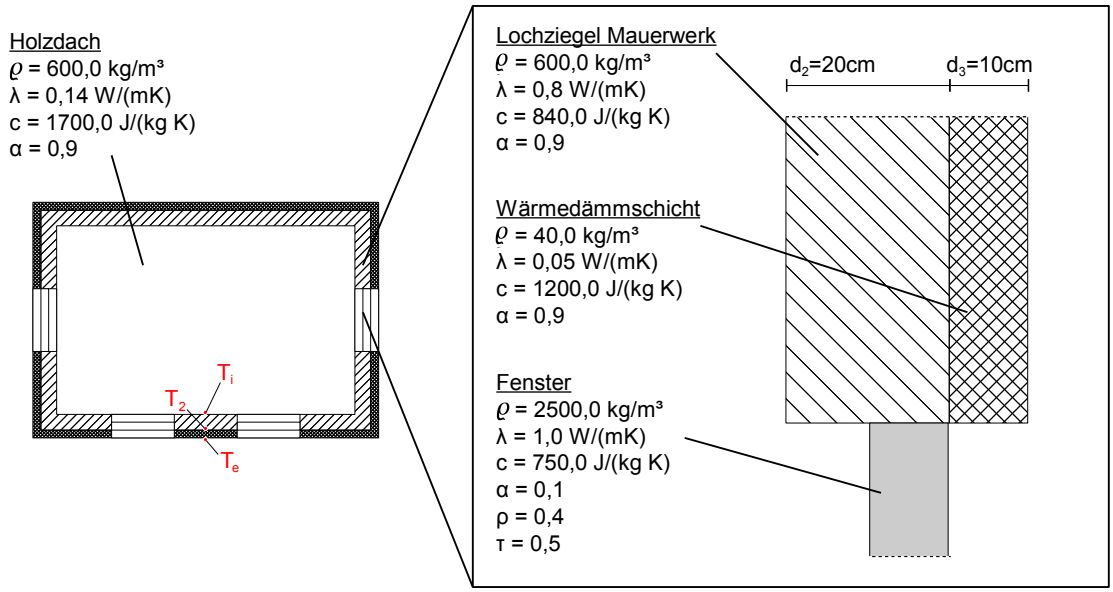


Abbildung 6.17: *Gebäudeaufbau und Materialeigenschaften eines Wohnhauses*

Die Simulation wird über zwei Tage mit einer sich ändernden solaren Einstrahlung (in Abhängigkeit des tatsächlichen Sonnenstandes) durchgeführt. Die diffuse und direkte Solarstrahlung, sowie die zugehörigen Höhen- h und Vertikalwinkel v sind einem Wetterdatensatz der DIN 4710 [33] für den Standort Berlin im Monat Juli entnommen. Die verwendeten Sonnenstände sind in Abbildung 6.18 dargestellt. Zwischen den fünf Positionen der Sonne wird bei der Strahlungsberechnung linear interpoliert.

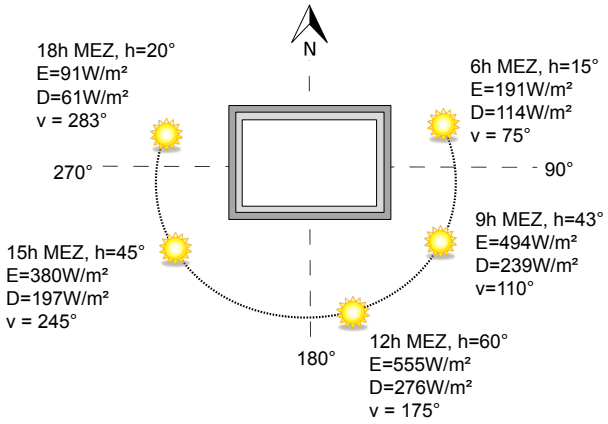


Abbildung 6.18: *Sonnenstände mit diffuser und direkter Strahlung*

Als Randbedingungen werden an allen Bauteiloberflächen Neumann-Randbedingungen verwendet. Die Anfangstemperatur der Bauteile und Umgebung ist mit 20°C angenommen. Das Gebäude besteht aus 46 einzelnen Bauteilen, die über 47 Interfaces miteinander verbunden sind. Für die Strahlungsberechnung wird das Oberflächennetz mit ca. 78.000 Dreiecken diskretisiert. Die Strukturgitter der Bauteile umfassen insgesamt ca. 850.000 Gitterknoten für die FDM-Berechnung. Das System wird über 24 Stunden vorinitialisiert, so dass sich sinnvolle Temperaturen in den Bauteilen einstellen. Die im Folgenden dargestellten Ergebnisse beziehen sich auf den zweiten Tag der Simulation. Abbildung 6.19 zeigt die Energieverteilung am Gebäude für verschiedene Uhrzeiten.

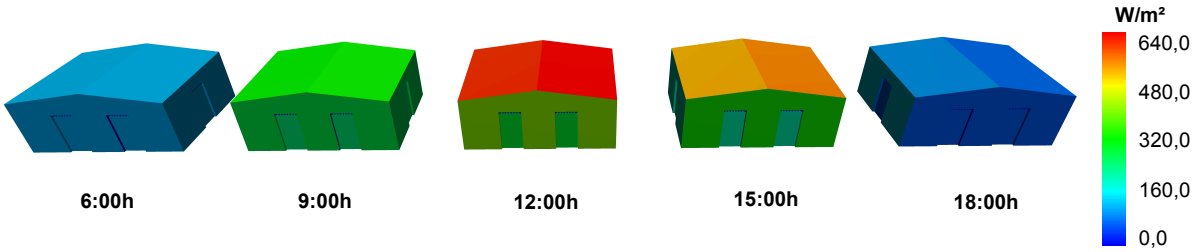


Abbildung 6.19: *Energieverteilung durch direkte und diffuse Solarstrahlung*

In Abbildung 6.20 ist die Temperaturverteilung am Gebäude um 12:00 Uhr in mehreren horizontalen und vertikalen Schnitten aufgeführt. Die Laufzeit für diese Simulation beträgt ca. 1.700 Sekunden bei ca. 65MB Speicherbedarf auf einem Intel Core2 Q9550 Quad-Core. Die zeitliche Temperaturentwicklung in der Südfassade des Gebäudes für die Punkte T_i, T_2 und T_e (vgl. Abbildung 6.17) ist in Abbildung 6.21 gegeben. Zu Erkennen ist die zeitlich verzögerte und abgeschwächte Temperaturänderung zwischen Wärmedämmschicht und Mauerwerk sowie an der inneren Bauteiloberfläche.

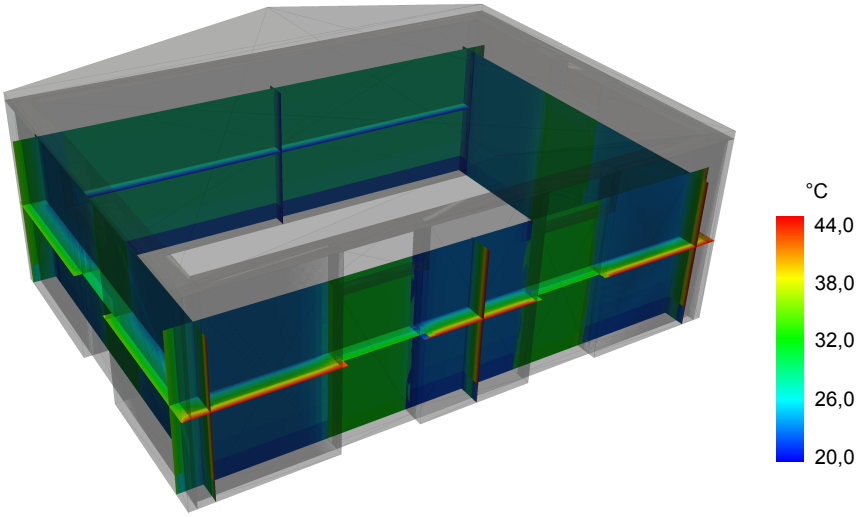


Abbildung 6.20: *Temperaturverteilung in einem Wohnhaus*

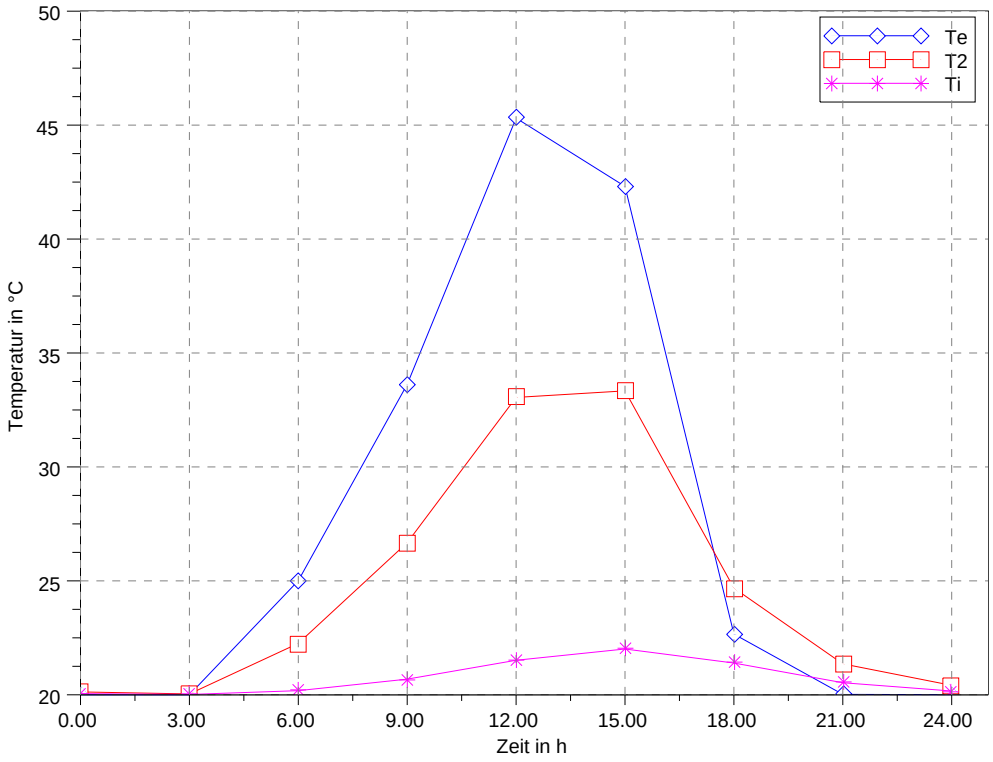


Abbildung 6.21: *Zeitliche Temperaturentwicklung in der Südfassade des Wohnhauses*

6.2.5 Thermische Untersuchung einer Doppelfassade

Zur Simulation des voll gekoppelten Problems (Strahlung, Wärmeleitung und Konvektion) wurden bereits erste Versuche unternommen. Hierzu wird ein bestehender CFD-Code [1] auf Basis der Lattice-Boltzmann-Methode (LBM) an den Strahlungs-Struktur-Kernel gekoppelt [16, 17, 69]. Die LB-Methode hat sich zur Simulation von thermischen Strömungen in komplexen Geometrien etabliert und lässt sich aufgrund ihrer Struktur besonders effizient parallel (auf Computerclustern oder Grafikkarten) ausführen. In dem hier aufgeführten Beispiel wird die thermische Untersuchung einer Doppelfassade durchgeführt. Ziel dieser Simulation ist die Optimierung der Gebäudestruktur eines Bürokomplexes in der Planungsphase für verschiedene bauliche Ausführungen. Abbildung 6.22 zeigt das 3D-Modell der Fassade, die Geschwindigkeit und die Temperaturverteilung aufgrund einer konstanten solaren Einstrahlung von $500 W/m^2$ bei einer Anfangstemperatur von 20°C. Die Strömungssimulation beansprucht hierbei ca. 90% der Rechenzeit. Die Kopplung der drei Wärmetransportprozesse ist noch nicht vollständig umgesetzt und erfordert weitere Forschungsarbeit.

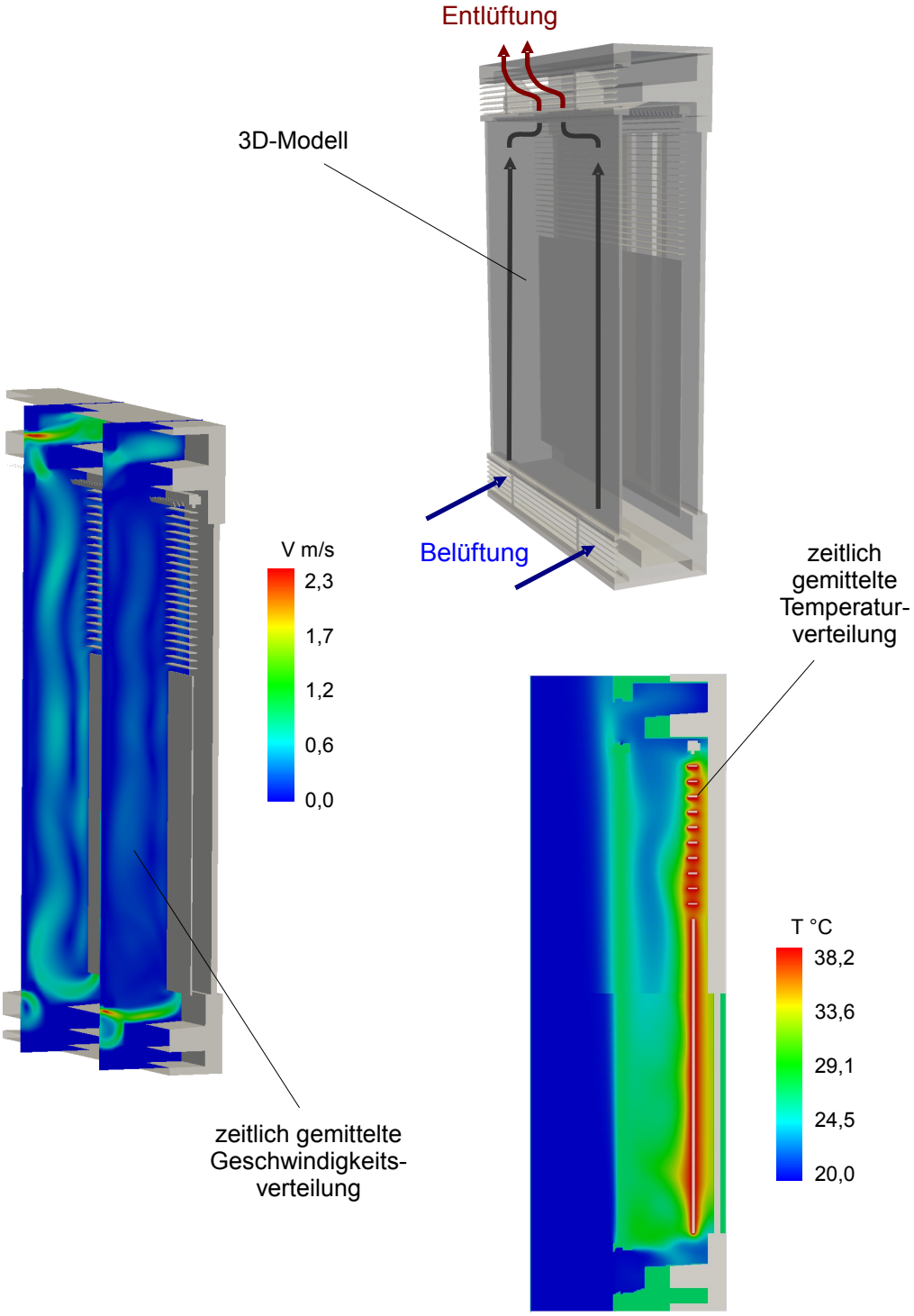


Abbildung 6.22: *Thermische Untersuchung einer Doppelfassade*

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

Der im Rahmen dieser Arbeit vorgestellte Software-Prototyp eignet sich prinzipiell zur Simulation thermischer Phänomene in komplexen Gebäuden und unterstützt Fachplaner und Ingenieure bei der virtuellen Optimierung und Planung von Gebäudestrukturen. Hierbei werden effiziente numerische Ansätze für die thermischen Berechnungen entwickelt, die eine interaktive Simulation mit schnellen Antwortzeiten ermöglichen. Die verwendete Radiosity-Methode zur Berechnung des Strahlungsaustausches ist um eine adaptive Verfeinerung des Oberflächennetzes und einen Ansatz zur Berücksichtigung von Transmissionseffekten erweitert. Außerdem wird zur weiteren Beschleunigung des Verfahrens die Geometrie in einem optimierten Kd-Baum zur Raumpartitionierung abgelegt. Durch diese Ansätze kann die algorithmische Komplexität des Strahlungsaustausches deutlich reduziert werden. Die angekoppelte Berechnung der Wärmeleitung in der Struktur basiert auf der Finite-Differenzen-Methode, welche hardwarebeschleunigt auf Grafikkarten ausgeführt wird und zu einer signifikanten Steigerung der Performance führt. Durch Einsatz einer effizienten Datenstruktur zur Speicherung von komplexen Gebäudemodellen, erweitert um Materialparameter und Randbedingungen auf Basis eines einheitlichen IFC-Gebäudedatenmodells, wird ein softwareübergreifender Datenaustausch ermöglicht.

Als Grundlage des virtuellen Entwurfsraums zur Erstellung von Gebäudemodellen dient AutoCAD Architecture, das direkt an die laufende Simulation gekoppelt ist. AutoCAD ist um zahlreiche Funktionalitäten erweitert u.a. für das Pre-processing, den Datenaustausch, zur Ereigniserkennung und Grenzwertüberwachung sowie zur Aufbereitung des Datenmodells und kann somit direkt als Planungswerkzeug zur Optimierung von Bauwerken unter thermischen Aspekten verwendet werden. Durch Einsatz von Computational Steering Techniken können Planer und Ingenieure interaktiv, d.h. zur Laufzeit der Simulation, Ergebnisse analysieren und das System steuern, um beispielsweise Änderungen an der Gebäudegeometrie vorzunehmen. Zur Visualisierung der in der Simulation in jedem Zeitschritt anfallenden Datenmengen wird ein neuer Ansatz vorgestellt, der neben der parallelen Simulation auch die Visualisierung verteilt auf einem Computercluster durchführt. Hierbei werden die Ergebnisse der Simulation direkt auf den Clusterknoten gerendert, so dass keine zusätzliche Datenkommunikation über ein Netzwerk notwendig ist. Durch diesen Ansatz wird es möglich, die in der thermischen Simulation in jedem Zeitschritt anfallenden großen Datenmengen in Echtzeit zu visualisieren und interaktiv in die Simulation einzugreifen. Die Visualisierung erfolgt hierbei in einer Tiled-Display-Umgebung, die eine ideale Plattform für kooperative

Planungsprozesse darstellt und es einer Gruppe von Fachplanern erlaubt, interaktiv Gebäude unter thermischen Gesichtspunkten zu optimieren und in die Gebäudeplanungsprozesse einzubeziehen.

Abschließend wird anhand zahlreicher Validierungen gezeigt, dass die erzielten numerischen Fehler des entwickelten Simulationskerns gegenüber dem physikalischen Modellfehler gering sind und somit zur Untersuchung bauphysikalischer Problemstellungen sehr gut geeignet ist.

7.2 Ausblick

Die prototypische Umsetzung der thermischen Simulationsumgebung liefert bereits sehr gute Ergebnisse für praxisrelevante Anwendungen aus dem Bauingenieurwesen. Allerdings besteht bei einigen Verfahren noch Optimierungsbedarf.

So wird die Parallelisierung der hierarchischen Radiosity-Methode noch nicht ausreichend betrachtet. Für das klassische Radiosity Verfahren wurden zahlreiche Ansätze zur parallelen Berechnung auf Shared und Distributed Shared Memory Systemen entwickelt, die häufig auf einer gleichmäßigen Aufteilung der Iterationen (Energietransfer von einem Emitter an eine Untermenge von Oberflächenelementen) auf die einzelnen Prozesse basieren [44]. Diese Ansätze sind für die verteilte Berechnung der hierarchischen Radiosity-Methode nicht geeignet. Schwierig gestaltet sich die sinnvolle Lastverteilung, die aufgrund der adaptiven Verfeinerung des Oberflächennetzes während der Berechnung nicht a priori festgelegt werden kann. Hierbei ist die adaptive Strahlungsberechnung um Verfahren zur dynamischen Lastverteilung zur Laufzeit der Simulation zu erweitern.

Weiterer Forschungsbedarf besteht in der automatisierten Erstellung von Bauteil-Interfaces (zur Realisierung des Wärmeübergangs zwischen aneinandergrenzenden Bauteilen). Interfaces werden bislang nur für geometrisch einfache Anordnungen selbstständig angelegt. Hier ist eine automatische Erstellung für Bauteile beliebiger Lage und Orientierung zu entwickeln. Insbesondere die Erstellung von Interfaces zwischen Bauteilen mit komplexen gekrümmten Oberflächen und zahlreichen geometrischen Sonderfällen, ist nicht trivial. Des Weiteren muss, bei solchen Geometrien, das Auffinden gültiger Nachbarpunkten für die Interpolationen an den Interfaces erweitert werden.

Ein weiterer Punkt der noch nicht betrachtet wurde ist das Load-balancing bei der Berechnung der Wärmeleitung verteilt auf mehreren GPUs und CPUs. Hier könnten Ansätze aus der verteilten Strömungsmechanik adaptiert werden. Zum Beispiel konnte in [101] gezeigt werden das mit einer effizienten Lastverteilung auch über Cluster von Grafikkarten sehr gute Performance (Speedup/Effizienz) erreicht werden kann.

Die Erweiterung des hier vorgestellten Simulationskerns um Verfahren zur Berechnung von Gasstrahlung würde die Simulation von Verbrennungsprozessen, Bränden und der Ausbreitung von Rauchgas ermöglichen. Durch die in diesen Prozessen auftretenden hohen Tempe-

raturen darf dabei jedoch der Einfluss des Zwischenmediums auf die Strahlungsausbreitung nicht vernachlässigt werden. Das Medium wirkt hierbei, anders als bei niedrigen Temperaturen, absorbierend, emittierend und streuend auf die Strahlung. Die numerische Berechnung von Gasstrahlung in komplexen Umgebungen kann mit unterschiedlichen Verfahren wie der Finite-Volumen-Methode oder der Monte-Carlo-Simulation erfolgen [105] und ließe sich direkt in die Strukturen der entwickelten Simulationsumgebung einbinden.

Weiterhin ist es denkbar für die Konstruktion von Gebäuden den frei erhältlichen Geometriemodellierer Google SketchUp [47] zu verwenden. Dieser ließe sich über die Entwicklung eines entsprechenden Plug-ins direkt in die Simulationsumgebung integrieren. Der große Vorteil liegt neben den sehr einfach zu erlernenden Zeichenfunktionalitäten darin, dass sich komplexe Stadtmodelle aus Google Earth [46] direkt für Simulationen verwenden ließen. Somit kann z.B. bei der Planung und Optimierung von Solar- und Photovoltaikanlagen oder der Tageslichtsimulation auch die Verschattung von Nachbargebäuden leicht berücksichtigt werden.

Des Weiteren ist die Kopplung der Strahlungs-Struktur-Simulation an einen Strömungslöser zur Berechnung der Konvektion geplant. Hierzu wurden bereits erste Versuche zur Kopplung an einen bestehenden CFD-Code auf Basis der Lattice-Boltzmann-Methode (LBM) unternommen (vgl. hierzu Abschnitt 6.2.5). Die Kopplung der drei Wärmetransportprozesse ist noch nicht vollständig umgesetzt, hier ist weiterer Forschungsbedarf notwendig. Größtes Problem stellt die Kopplung der unterschiedlichen zeitlichen Skalen der einzelnen Transportprozesse dar. Strahlungsaustausch findet auf der Sekundenskala, Konvektion im Bereich von Zehntelsekunden und Wärmeleitung in der Struktur im Bereich von Stunden statt. Hier ist es denkbar Mittelungsansätze oder Homogenisierungsverfahren zu verwenden.

Die Integration von Optimierungsverfahren in die entwickelte Simulationsumgebung könnte Ingenieure bei der Planung von Gebäuden unterstützen. Hier ließen sich durch die Sensitivitätsanalyse von Parametern vorgegebene Zielwerte annähern und somit z.B. die unter thermischen Gesichtspunkten optimalen Materialeigenschaften einer Konstruktion finden oder der Sonnenschutz einer Fassade optimal einstellen. Die physikalisch algorithmische Optimierung von Parametern kann durch die Analyse von Modellgleichungen oder Iterativ z.B. auf Grundlage von evolutionären Algorithmen erfolgen.

Literaturverzeichnis

[1] Ahrenholz, B.: *Massively parallel simulations of multiphase- and multicomponent flows using lattice Boltzmann methods.* Dissertation, Technischen Universität Carolo-Wilhelmina zu Braunschweig, 2009.

[2] Ahrens, J. P., Li, K. und Reed, D. A.: *Next-generation visualization displays: the research challenges of building tiled displays (panel session).* In: *VIS '00: Proceedings of the conference on Visualization '00*, S. 527–529, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[3] Akenine-Möller, T., Haines, E. und Hoffman, N.: *Real-Time Rendering.* A. K. Peters, Ltd., Natick, MA, USA, 3. Aufl., 2008.

[4] AMD: *ATI Stream Technologie*, 2010. `http://www.amd.com/us/products/technologies/stream-technology/Pages/stream-technology.aspx`; eingesehen am 30. Juli 2010.

[5] Autodesk: *OMF Developer's Guide*, 2002. `http://adn.autodesk.com`; eingesehen am 30. Juli 2010.

[6] Autodesk: *ObjectARX Developer's Guide*, 2008. `http://adn.autodesk.com`; eingesehen am 30. Juli 2010.

[7] Autodesk AutoCAD, 2010. `http://www.autodesk.com`; eingesehen am 30. August 2010.

[8] Badouel, D.: *An efficient ray-polygon intersection.* Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[9] Baehr, H. und Stephan, K.: *Wärme- und Stoffübertragung.* Springer Verlag, 5. Aufl., 2006.

[10] Barnes, J. und Hut, P.: *A hierarchical O(N log N) force calculation algorithm.* Nature, 324:446–449, 1986.

[11] Böckh, P. von und Wetzel, T.: *Wärmeübertragung: Grundlagen und Praxis* . Springer, Berlin, 2009.

[12] Bender, M. und Brill, M.: *Computergrafik: Ein anwendungsorientiertes Lehrbuch.* Hanser Fachbuchverlag, 2005.

[13] Bentley, J. L.: *Multidimensional binary search trees used for associative searching.* Communications of the ACM, 18(9):509–517, 1975.

[14] Berg, M. de, Cheong, O., Kreveld, M. van und Overmars, M.: *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3. Aufl., 2008.

[15] Bindick, S.: *Ein interaktiver virtueller Entwurfsraum zur thermischen Simulation im Rahmen kooperativer Planungsprozesse*. In: Krämer, T., Richter, S., Enge, F. und Kraft, B. (Hrsg.): *Forum Bauinformatik*, Bd. 9, S. 23–28. Book Series of the Department of Civil Engineering, TU Berlin, 2010.

[16] Bindick, S., Ahrenholz, B. und Krafczyk, M.: *Transiente 3D-Simulation thermisch induzierter Strömungen in und an Gebäuden mit kinetischen Methoden*. In: Peil, U. (Hrsg.): *WtG-Berichte Nr. 11 - Windingenieurwesen in Forschung und Praxis*, S. 215–220, 2009.

[17] Bindick, S., Ahrenholz, B. und Krafczyk, M.: *Heat Transfer*, Kap. Efficient Simulation of Transient Heat Transfer Problems in Civil Engineering. INTECH Open Access Publisher, 2011. ISBN: 978-953-307-550-1.

[18] Bindick, S. und Nachtwey, B.: *Implementierung eines parallelen Radiosityverfahrens zur Berechnung von strahlungsbeeinflussten Oberflächentemperaturen*. In: Merkel, I. A. P., Schütz, R. und Wießflecker, T. (Hrsg.): *Forum Bauinformatik 2007*, S. 11–18. Verlag der Technischen Universität Graz, 2007.

[19] Bindick, S., Stiebler, M. und Krafczyk, M.: *Fast kd-tree based hierarchical radiosity for radiative heat transport problems*. International Journal for Numerical Methods in Engineering, 2010. akzeptiert zur Veröffentlichung.

[20] Bindick, S., Stiebler, M. und Krafczyk, M.: *Interaktive Simulation transienter Wärmetransportprozesse in und an Gebäuden*. In: Mahdavi, A. und Martens, B. (Hrsg.): *Building Performance Simulation in a Changing Environment - Proceedings of the Third German-Austrian IBPSA Conference - Vienna University of Technology*, S. 33–38, 2010.

[21] Bohn, C. A. und Garmann, R.: *A Parallel Approach to Hierarchical Radiosity*. In: *University of West Bohemia*, S. 26–35, 1995.

[22] Bryson, S.: *Virtual environments in scientific visualization*. In: *VRST '94: Proceedings of the conference on Virtual reality software and technology*, S. 201–220, River Edge, NJ, USA, 1994. World Scientific Publishing Co., Inc.

[23] Bryson, S.: *Virtual reality in scientific visualization*. Communications of the ACM, 39(5):62–71, 1996.

[24] buildingSMART e.V. (2010). `http://www.buildingsmart.de/`; eingesehen am 30. Juli 2010.

[25] Burtsev, S. V. und Kuzmin, Y. P.: *An efficient flood-filling algorithm*. Computers & Graphics, 17(5):549–561, 1993.

[26] Calit2 - California Institute for Telecommunications and Information Technology. `http://www.calit2.net/`; eingesehen am 30. Juli 2010.

[27] CGLX: *(Cross-Platform Cluster Graphic Library), California Institute for Telecommunications and Information Technology (Calit2) at University of California, San Diego*, 2010. `http://vis.ucsd.edu/~cglx/`; eingesehen am 30. Juli 2010.

[28] Cohen, M. F., Chen, S. E., Wallace, J. R. und Greenberg, D. P.: *A progressive refinement approach to fast radiosity image generation.* SIGGRAPH Computer Graphics, 22(4):75–84, 1988.

[29] Cohen, M. F. und Greenberg, D. P.: *The hemi-cube: a radiosity solution for complex environments.* In: *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, S. 31–40, New York, NY, USA, 1985. ACM.

[30] Cohen, M. F., Wallace, J. und Hanrahan, P.: *Radiosity and realistic image synthesis.* Academic Press Professional, Inc., San Diego, CA, USA, 1993.

[31] DeFanti, T., Leigh, J., Renambot, L., Jeong, B., Verlo, A., Long, L., Brown, M. Sandin, D., Vishwanath, V., Liu, Q., Katz, M., Papadopoulos, P., Keefe, J., Hidley, G., Dawe, G., Kaufman, I., Glogowski, B., Doerr, K., Singh, R., Girado, J., Schulze, J., Kuester, F., und Smarr, L.: *The OptIPortal, a Scalable Visualization, Storage, and Computing Interface Device for the OptiPuter.* Future Generation Computer Systems, Elsevier, 25:114–123, 2009.

[32] DIN 1349: *Durchgang optischer Strahlung durch Medien.* Berlin: Beuth Verlag, Juni 1972.

[33] DIN 4710: *(Statistiken meteorologischer Daten zur Berechnung des Energiebedarfs von heiz- und raumlufttechnischen Anlagen in Deutschland)*, 2003.

[34] DIN 9288: *Wärmeschutz - Wärmeübertragung durch Strahlung - Physikalische Größen und Definitionen*, 1996.

[35] DIN EN ISO 6946: *Wärmedurchlasswiderstand und Wärmedurchgangskoeffizient*, 2008.

[36] Doerr, K. und Kuester, F.: *CGLX: A Scalable, High-performance Visualization Framework for Networked Display Environments.* IEEE Transactions on Visualization and Computer Graphics, 99:1077–2626, 2010.

[37] Durand, F., Drettakis, G. und Puech, C.: *Fast and accurate hierarchical radiosity using global visibility.* ACM Transactions on Graphics, 18(2):128–170, 1999.

[38] Fahrig, T.: *Kooperative Optimierung von Raumluftströmungen mittels agentengestützter Regelungstechnik in einer Computational Steering Umgebung.* Dissertation, Technischen Universität Carolo-Wilhelmina zu Braunschweig, 2007.

[39] Fischer, H. M., Jenisch, R., Stohrer, M., Homann, M., Freymuth, H., Richter, E. und Häupl, P.: *Lehrbuch der Bauphysik Schall, Wärme, Feuchte, Licht, Brand, Klima.* Viweg+Teubner Verlag, 6. Aufl., 2008.

[40] Fishkin, K. P. und Barsky, B. A.: *A family of new algorithms for soft filling.* S. 235–244, 1984.

[41] Foley, J. D., Dam, A. van, Feiner, S. K. und Hughes, J. F.: *Computer graphics: principles and practice (2nd ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[42] Fujimoto, A., Tanaka, T. und Iwata, K.: *Arts: Accelerated ray-tracing system.* IEEE Computer Graphics and Applications, 6(4):16–26, 1986.

[43] Georgia Institute of Technology: *Large Geometric Models Archive*, 2010. `http://www.cc.gatech.edu/projects/large_models/`; eingesehen am 09. September 2010.

[44] Gibson, S. und Hubbold, R. J.: *A Perceptually-Driven Parallel Algorithm for Efficient Radiosity Simulation.* IEEE Transactions on Visualization and Computer Graphics, 6(3):220–235, 2000.

[45] Glassner, A. S.: *Space Subdivision For Fast Ray Tracing.* IEEE Computer Graphics and Applications, 4(10):15–22, 1984.

[46] Google Earth, 2010. `http://www.google.com/earth`; eingesehen am 25. Oktober 2010.

[47] Google SketchUp, 2010. `http://sketchup.google.com/`; eingesehen am 25. Oktober 2010.

[48] Goral, C. M., Torrance, K. E., Greenberg, D. P. und Battaile, B.: *Modeling the interaction of light between diffuse surfaces.* SIGGRAPH Computer Graphics, 18(3):213–222, 1984.

[49] Graphisoft ArchiCAD, 2010. `http://www.graphisoft.de`; eingesehen am 30. Juli 2010.

[50] Haines, E.: *Essential ray tracing algorithms.* Academic Press Ltd., London, UK, 1989.

[51] Hairer, E., Nørsett, S. P. und Wanner, G.: *Solving Ordinary Differential Equations I: Non-stiff Problems (Springer Series in Computational Mathematics) (v. 1)*. Springer, 2. Aufl., 2009.

[52] Hanrahan, P., Salzman, D. und Aupperle, L.: *A rapid hierarchical radiosity algorithm.* In: *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, S. 197–206, New York, NY, USA, 1991. ACM.

[53] Havran, V.: *Heuristic Ray Shooting Algorithms.* Ph.D. Thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, 2000.

[54] Havran, V. und Bittner, J.: *On Improving KD-Trees for Ray Shooting.* Journal of WSCG, 10(1):209–216, 2002.

[55] Hottel, H. C. und Cohen, E. S.: *Radiant heat exchange in a gas-filled enclosure: allowance for non-uniformity of gas temperature.* AIChE Journal, 4:3–14, 1958.

[56] Hottel, H. C. und Mangelsdorf, H. G.: *Heat transmission by radiation from non-luminous gases.* Trans. Amer. Inst. Chem. Eng., 31:517–549, 1935.

[57] Howell, J. R.: *A Catalog of Radiation Configuration Factors.* McGraw-Hill, New York, 1982.

[58] IAI - Industrieallianz für Interoperabilität e.V.: *Anwenderhandbuch Datenaustausch BIM/IFC.* buildingSMART, München 2008.

[59] IFC Spezifikation der buildingSMART, 2010. `http://www.iai-tech.org`; eingesehen am 30. Juli 2010.

[60] IFC Zertifizierung, 2010. `http://www.buildingsmart.de/2/2_01_03.htm`; eingesehen am 30. Juli 2010.

[61] Incropera, F. P.: *Fundamentals of Heat and Mass Transfer.* John Wiley & Sons, 2006.

[62] Johnson, R. W.: *The Handbook of Fluid Dynamics.* CRC Press, 1. Aufl., 1998.

[63] Karlsruhe Institute of Technology (KIT): *IfcViewer*, 2010. `http://www.iai.fzk.de/www-extern/index.php?id=1138`; eingesehen am 30. Juli 2010.

[64] Kay, T. L. und Kajiya, J. T.: *Ray tracing complex scenes.* In: *SIGGRAPH*, S. 269–278, 1986.

[65] Kühner, S.: *Virtual Reality basierte Analyse und interaktive Steuerung von Strömungssimulationen im Bauingenieurwesen.* Dissertation, Technische Universität München, 2003.

[66] Khronos Group: *OpenCL - The open standard for parallel programming of heterogeneous systems*, 2010. `http://www.khronos.org/opencl/`; eingesehen am 30. Juli 2010.

[67] Kirk, D. B. und Hwu, W. m. W.: *Programming Massively Parallel Processors: A Hands-on Approach.* Morgan Kaufmann, 1. Aufl., 2010.

[68] Krafczyk, M., Tölke, J. und Fahrig, T.: *Vernetzt-kooperative Planungsprozesse im Konstruktiven Ingenieurbau*, Kap. Ein Prototyp für verteilte, interaktiv-kooperative Simulationen zur Beschleunigung von Entwurfszyklen im Konstruktiven Ingenieurbau. Springer Verlag, 2007.

[69] Krafczyk, M., Tölke, J., Ahrenholz, B., Bindick, S., Freudiger, S., Geller, S., Janßen, C. und Nachtwey, B.: *100 Volumes of 'Notes on Numerical Fluid Mechanics'*, Kap. Kinetic Modeling and Simulation of Environmental and Civil Engineering Flow Problems, S. 341–350. Springer-Verlag, Berlin, 2009.

[70] Kramer, B.: *ObjectARX Primer (Autodesk's Programmer).* Delmar Cengage Learning, 1999.

[71] Kuhlmann, H. C.: *Strömungsmechanik.* PEARSON STUDIUM, 1. Aufl., 2007.

[72] Li, Z. und Varshney, A.: *A Real-Time Seamless Tiled Display for 3D Graphics.* In: *Proceedings, Seventh Annual Symposium on Immersive Projection Technology (IPT 2002)*, 2002.

[73] Lin, Z.: *An Automatic Mesh Generator for a Computational Steering Framework.* Studienarbeit, Technische Universität Braunschweig, 2006.

[74] Linxweiler, J., Krafczyk, M. und Tölke, J.: *Highly interactive computational steering for coupled 3D flow problems utilizing multiple GPUs.* Computing and Visualization in Science, Springer Berlin / Heidelberg, S. 1–16, 2011.

[75] MacDonald, D. J. und Booth, K. S.: *Heuristics for ray tracing using space subdivision.* The Visual Computer: International Journal of Computer Graphics, 6(3):153–166, 1990.

[76] Maple (mathematical manipulation language) - Maplesoft: *Computeralgebrasystem*, 2010. `http://www.maplesoft.com/products/Maple`; eingesehen am 30. Juli 2010.

[77] McAuley, C.: *Programming AutoCAD Using ObjectARX (Autodesk's Programmer).* Cengage Learning Services, 2000.

[78] McCormick, B. H., DeFanti, T. A. und (ed), M. D. B.: *Visualization in Scientific Computing.* ACM SIGGRAPH, New York, 1987.

[79] Modest, M. F.: *Radiative Heat Transfer.* Academic Press, An imprint of Elsevier Science, 2. Aufl., 2003.

[80] Möller, T. und Trumbore, B.: *Fast, Minimum Storage Ray-Triangle Intersection.* journal of graphics, gpu, and game tools, 2(1):21–28, 1997.

[81] Mulder, J., Wijk, J. van und Liere, R. van: *A survey of computational steering environments.* Future generation computer systems, Vol. 15(2):119–129, 1999.

[82] Nemetschek Allplan, 2010. `http://www.nemetschek.de`; eingesehen am 30. Juli 2010.

[83] Neuberg, F.: *Ein Softwarekonzept zur Internet-basierten Simulation des Ressourcenbedarfs von Bauwerken.* Dissertation, Technische Universität München, 2003.

[84] Nvidia: *Compute Unified Device Architecture (CUDA)*, 2010. `http://developer.nvidia.com/object/cuda_3_1_downloads.html`; eingesehen am 30. Juli 2010.

[85] Nvidia: *Nvidia CUDA Programming Guide 3.1*, 2010.

[86] Ogayar, C. J., Rueda, A. J., Segura, R. J. und Feito, F. R.: *Fast and simple hardware accelerated voxelizations using simplicial coverings.* The Visual Computer: International Journal of Computer Graphics, 23(8):535–543, 2007.

[87] Ogayar, C. J., Segura, R. J. und Feito, F. R.: *Point in solid strategies.* Computers & Graphics, 29(4):616 – 624, 2005.

[88] OpenSceneGraph (OSG), 2009. `http://www.openscenegraph.org/`; eingesehen am 30. Juli 2010.

[89] O'Rourke, J.: *Computational Geometry in C.* Cambridge University Press, 1. Aufl., 1994.

[90] Parker, S. G., Johnson, C. R. und Beazley, D.: *Computational Steering Software Systems and Strategies.* IEEE Computing in Science & Engineering, 4(4):50–59, 1997.

[91] Pellegrini, M.: *Monte Carlo approximation of form factors with error bounded a priori.* In: *SCG '95: Proceedings of the eleventh annual symposium on Computational geometry*, S. 287–296, New York, NY, USA, 1995. ACM.

[92] Pharr, M. und Humphreys, G.: *Physically Based Rendering: From Theory to Implementation.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[93] Pianykh, O. S., Tyler, J. M. und Jr., W. N. W.: *Improved Monte Carlo form factor integration.* Computers & Graphics, 22(6):723–734, 1998.

[94] Podehl, A., Rauber, T. und Rünger, G.: *A shared-memory implementation of the hierarchical radiosity method.* Theoretical Computer Science, 196(1-2):215–240, 1998.

[95] Polifke, W. und Kopitz, J.: *Wärmeübertragung - Grundlagen, analytische und numerische Methoden*, Bd. 2. Pearson Studium, 2009.

[96] Press, W., Teukolsky, S., Vetterling, W. und Flannery, B.: *Numerical Recipes in C.* Cambridge University Press, Cambridge, UK, 2nd Aufl., 1992.

[97] Reddy, J. N. und Gartling, D. K.: *The Finite Element Method in Heat Transfer and Fluid Dynamics.* Crc Pr Inc, 3. Aufl., 2010.

[98] Rueda, A. J., Segura, R. J., Feito, F. R., de Miras, J. R. und Ogayar, C. J.: *Voxelization of Solids Using Simplicial Coverings.* In: *WSCG (Short Papers)*, S. 227–234, 2004.

[99] Sanders, J. und Kandrot, E.: *CUDA by Example: An Introduction to General-Purpose GPU Programming.* Addison-Wesley Professional, 1. Aufl., 2010.

[100] Schäfer, S.: *Efficient Object-Based Hierarchical Radiosity Methods.* Dissertation, Technische Universität Braunschweig, 2000.

[101] Schönherr, M., Kucher, K. und Krafczyk, M.: *SKALB - (Lattice-Boltzmann-Methoden für skalierbare Multi-Physik-Anwendungen) - Zwischenbericht 3*, 2010. `http://www.skalb.de`; eingesehen am 30. Juli 2010.

[102] Shani, U.: *Filling regions in binary raster images: A graph-theoretic approach.* SIGGRAPH Computer Graphics, 14:321–327, July 1980.

[103] Shaw, E.: *Hierarchical Radiosity for Dynamic Environments.* Computer Graphics Forum, 16:107–118, 1997.

[104] Shevtsov, M., Soupikov, A. und Kapustin, A.: *Highly Parallel Fast KD-tree Construction for Interactive Ray Tracing of Dynamic Scenes.* Computer Graphics Forum, 26(3):395–404, 2007.

[105] Siegel, R. und Howell, J. R.: *Thermal radiation heat transfer.* Taylor & Francis Inc, 3. Aufl., 2002.

[106] Sinop, A. K., Abaci, T., Akkuş, U., Gürsoy, A. und Güdükbay, U.: *PHR: A Parallel Hierarchical Radiosity System with Dynamic Load Balancing.* The Journal of Supercomputing, 31(3):249–263, 2005.

[107] Stiller, A.: *Hyper-Ultra-Threader - Software-Entwicklung für GPUs im Rechenmodus.* ct - Magazin für Computertechnik, 20:190–195, 2009.

[108] Szécsi, L.: *Graphics programming methods*, Kap. An effective implementation of the k-D tree, S. 315–326. Charles River Media, Inc., Rockland, MA, USA, 2003.

[109] Szirmay-Kalos, L., Havran, V., Balázs, B. und Szécsi, L.: *On the Efficiency of Ray-shooting Acceleration Schemes.* In: Chalmers, A. (Hrsg.): *Proceedings of the 18th Spring Conference on Computer Graphics (SCCG 2002)*, S. 89–98, Budmerice, Slovakia, 2002. ACM Siggraph.

[110] Tampieri, F.: *Accurate form-factor computation.* Academic Press Professional, Inc., San Diego, CA, USA, 1992.

[111] TEKLA Corporation - Tekla Structures, 2010. `http://www.tekla.com`; eingesehen am 30. Juli 2010.

[112] Tölke, J., Fahrig, T., Nachtwey, B. und Krafczyk, M.: *Computational steering in civil engineering.* Proceedings of IKM Weimar, 2003.

[113] Trnsys, T., 2010. `http://www.transsolar.com`; eingesehen am 20. Oktober 2010.

[114] Tulke, J., Tauscher, E. und Theiler, M.: *Open IFC Tools*, 2010. `http://www.openifctools.org`; eingesehen am 30. Juli 2010.

[115] Vizelia Facility Online, 2010. `http://www.vizelia.com/en/index.php`; eingesehen am 30. Juli 2010.

[116] Wald, I.: *Realtime Ray Tracing and Interactive Global Illumination.* Dissertation, Computer Graphics Group, Saarland University, 2004.

[117] Wald, I. und Havran, V.: *On building fast kd-Trees for Ray Tracing, and on doing that in O(N log N).* In: *In proceedings of the 2006 IEEE symposium on interactive ray tracing, Salt Lake City, UTAH, USA*, S. 61–69, 2006.

[118] Wald, I., Mark, W. R., Günther, J., Boulos, S., Ize, T., Hunt, W., Parker, S. G. und Shirley, P.: *State of the Art in Ray Tracing Animated Scenes.* In: Schmalstieg, D. und Bittner, J. (Hrsg.): *STAR Proceedings of Eurographics 2007*, S. 89–116, Prague, Czech Republic, 2008. Eurographics Association.

[119] Wald, I., Slusallek, P., Benthin, C. und Wagner, M.: *Interactive Rendering with Coherent Ray Tracing.* In: Chalmers, A. und Rhyne, T. M. (Hrsg.): *Computer Graphics Forum*, Bd. 20, S. 153–164, 2001.

[120] Wallace, J. R., Elmquist, K. A. und Haines, E. A.: *A Ray tracing algorithm for progressive radiosity.* SIGGRAPH Computer Graphics, 23(3):315–324, 1989.

[121] Watt, A.: *3D-Computergrafik.* Pearson Studium, 2001.

[122] Welty, J. R., Wicks, C. E., Wilson, R. E. und Rorrer, G.: *Fundamentals of Momentum, Heat and Mass Transfer.* John Wiley & Sons, Inc., 2001.

[123] Wenisch, P.: *Computational Steering of CFD Simulations on Teraflop-Supercomputers*. Dissertation, Technische Universität München, 2008.

[124] Wright, H., Crompton, R., Kharche, S. und Wenisch, P.: *Steering and visualization: Enabling technologies for computational science*. Future Generation Computer Systems, 26(3):506 – 513, 2010.